UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE SISTEMAS INFORMÁTICOS

TRABAJO DE FIN DE GRADO

Agile Product Inception: Framework Analysis and Support

Autor
Daniel Olea Martín

Tutor
Agustín Yagüe Panadero

Madrid, julio 2014

Acknowledgements

To my mentor, Agustín Yagüe, for his seemingly infinite patience.

Abstract

In this dissertation, after testing that neither the definition of Agile methodologies, nor the current tools that support them, such as Scrum or XP, gave guidance for stages of software development prior to the definition of the first interaction of development; we proceeded to study the state of the art of Inception techniques, that is, techniques to deal with this early phase of the project, that would help guide its development.

From the analysis of these Inception techniques, we defined what we considered as the essential properties of an Inception framework. With that list at hand, it was found that no current Inception framework supported all the features, also, we found that it did not exist, either, any software application on the market that did it.

Finally, after checking the above gaps, we defined the Inception framework "Agile Incepti-ON", with all the practices necessary to meet the requirements specified above. In addition to this, a software application was developed to support the practices defined in the Inception framework, called "Agile Dojo".

Resumen

En este trabajo, tras la comprobación de que ni la definición de las metodologías Ágiles, ni las herramientas actuales que las soportan, como Scrum o XP, marcaban el camino a seguir en fases del desarrollo de software anteriores a la definición de la primera iteración de desarrollo; se procedió a estudiar el estado del arte en técnicas de Inception, esto es, técnicas para afrontar esa fase primigenia del proyecto, que ayudasen a guiar el desarrollo del proyecto.

Del análisis de dichas técnicas de Inception, se definieron lo que se consideran las propiedades indispensables de una herramienta de Inception. Con esta lista, se comprobó que ninguna herramienta actual de Inception soportaba todas las características y que no existía tampoco, ninguna aplicación software en el mercado que lo hiciese.

Finalmente, tras haber comprobado las anteriores carencias, se definió la herramienta de Inception “Agile Incepti-ON”, con todas las prácticas necesarias para cubrir las características anteriormente definidas, y por otro lado, se desarrolló una aplicación software para dar soporte a las prácticas definidas en la herramienta de Inception, llamada “Agile Dojo”.

Contents

Table Index

Figure Index

1. Objectives

The objectives of this dissertation are:

1. Study current techniques and methodologies of Agile Inception.
    1. Study the current frameworks of Agile Inception to understand the characteristics that conform each one.
    2. Obtain a well defined image of what Agile Inception is, whether it is important or not and why (or why not) it should be used in projects govern by Agile methodologies.
    3. Review Agile Software Management frameworks, like XP, Scrum or Kanban, to see if they provide guidance or not as of Agile Inception.
2. Study current tools for Agile Management
    1. Discover and analyse a set of different state-of-the-art Agile management tools.
    2. See what the coverage is for the Agile Inception phase in these tools.
3. Propose a framework to support the Inception phase and a tool to support that framework.
    1. Create a framework that supports the Agile Inception phase as the conclusions drawn from the aforementioned studies say.
    2. Program an application that supports the proposed framework.
    3. Select a tool that diligently takes care of managing a project based on Agile methodologies, and integrate the application with that tool.

2. Agile Inception

Agile Software Development

Agile software development refers to a new set of software development methodologies, that are based on principles of iterative development, with self-organizing cross-functional teams.

As of late, Agile methodologies have achieved a great momentum in the software development community. Companies have understood that for the vast majority of the projects they face, traditional methodologies are not adequate.

The key values[KentBeck 01] which define Agile methodologies are:

- ***Individuals and interactions*** *over processes and tools.*
- ***Working software*** *over comprehensive documentation*
- ***Customer collaboration*** *over contract negotiation.*
- ***Responding to change*** *over following a plan.*

From this four values arose the following twelve principles:

1. *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
2. *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
3. *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time-scale.*
4. *Business people and developers must work together daily throughout the project.*
5. *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
6. *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
7. *Working software is the primary measure of progress.*
8. *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
9. *Continuous attention to technical excellence and good design enhances agility.*
10. *Simplicity--the art of maximizing the amount of work not done--is essential.*
11. *The best architectures, requirements, and designs emerge from self-organizing teams.*
12. *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.*

Current Agile Frameworks

In order to have a set of practices to follow, thus being easier to adopt Agile methodologies, some Agile frameworks are commonly used, such as Scrum, XP or Kanban.

Scrum

Scrum is a set of good practices, that are applied regularly in order to make teams comply with the agile principles, so that they can achieve a better co-operative environment. These practices are more oriented towards project management than towards development tasks.

Within Scrum, projects are carried out in short, fixed temporary periods of time called sprints. Each iteration must provide the product owner with a valuable piece of software. Moreover, there are a set of rules that the development team and various stakeholders need to follow each sprint.

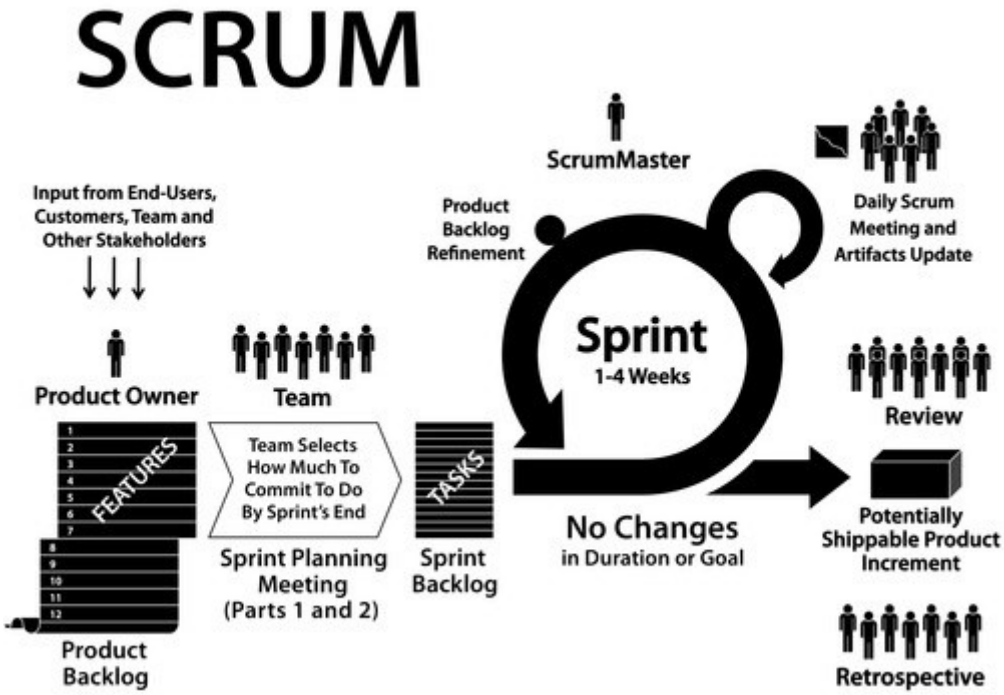


fig. i: Scrum process Via Ryan Connolly

The process starts with a prioritized list of objectives or requirements, written as user-stories, that serves as a lightweight, non-fixed, project plan. From that list, called backlog, the team selects a group of user-stories to complete in the upcoming sprint. Usually this selection is based on the ratio priority/effort, selecting the most valuable, less time-consumig user-stories.

Once the user-stories to be implemented are decided, the iteration, or sprint, starts. During the sprint, team members will carry out a daily, short, meeting, called daily scrum, that will serve as a synchronization point for the team.

When the sprint ends, the value is presented to the product owner, who reviews the pieces of software given and approves them or not.

Finally, at the end of the sprint, the team have a reunion in which they reflect about how things went.

Extreme Programming

Extreme programming (XP) is similar to Scrum, although in this case, XP practices are more focused on development than on project management. However, they share some common features.

fig. ii: Extreme Programming by Don Wells

XP also aims at delivering value in short iterations, although these are usually shorter than the Scrum's iterations. Iterations are planned by selecting user-stories from a repository.

Contrarily to Scrum, XP allows for user-stories to be swapped in mid-iteration, as long as the iteration remains the same as of workload, and that the user-story pulled out have not been started at that time.

Furthermore, XP teams deliver software work strictly following the order (priority) given by the customer, which is stored in the user-story repository. This is the major difference between Scrum and XP.

Finally, XP imposes some engineering practices like, pair programming, test-driven development and so forth, which Scrum does not.

Kanban

Kanban is another framework used for Agile software development. Kanban is meant to explicitly display in a board the work in progress (WIP), in order to pinpoint bottlenecks.

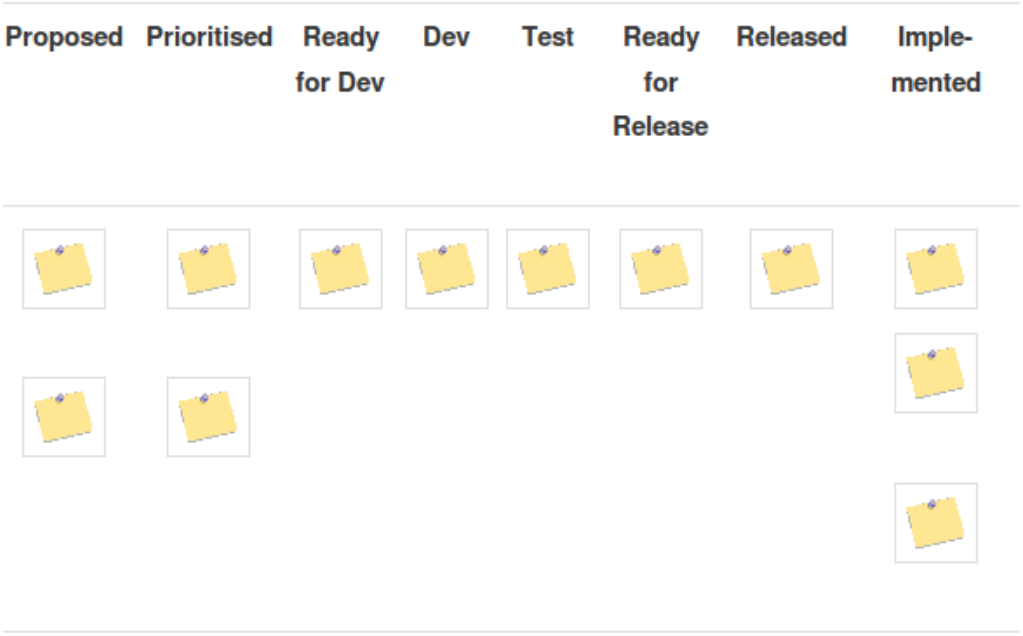


fig. iii: Kanban board by Mike Burrows

In Kanban, the concept of backlog is still present, but in this case it explicitly makes the team place the user-stories in a more detailed fashion, pinpointing the stage a certain story is in. By explicitly displaying the boundaries of the WIP in each stage, it is possible to identify imbalances in the development process.

It is, of course, iterative, offering value continuously to the customer. As the iteration progresses, the user-stories are pushed upwards in the pipeline of stages, until it is finally implemented.

Conclusions

Although mature enough to be used in real world projects, Agile frameworks are still in need of tuning. The original definition of the methodologies says nothing about how to face the start of a project. Thus, all these existent Agile frameworks start giving guidance from a filled backlog, neglecting previous states of the project, probably causing trouble to the team.

This lack of guidance was judged to be very damaging by some authors. One of these, Jonathan Rasmusson, was the one who coined the term “Agile Inception” and who defined what was necesary to be done, in his opinion, at the start of an Agile project, as a warm up for the team. However, Rasmusson was not alone, other authors also provided different solutions to try to mitigate problems that are rising within projects that start as the methodologies say, without an phase prior to the creation of the first iteration.

Brief Introduction to Agile Inception

Agile Inception is a proposed phase within Agile software development methodologies, which aims at populating the backlog and synchronizing team members towards a shared set of goals. Depending on the author, this could be achieved by stating a group of influencing factors such as risks, partners, etc.; by visually displaying the assumptions that lead to a goal; and so on.

However, it seems that all authors agree that this phase should always be carried, at least, at the start of a new project govern by Agile software methodologies. This agreement is based on the beneficial properties of the Agile Inception.

If done properly, the Inception phase should discover whether a project is doomed from its start and it should help the team understand the rationale under a set of tasks, allowing them to come up with better solutions.

In my personal opinion, based on my work experience, that last benefit is utterly essential for the well-being of a software project. Having a team of developers making software without clearly knowing its purpose, does not only discourage them and makes them less prone to creative solutions, but also, it usually leads to worse solutions, increasing the time needed to refactor the existing code afterwards, or decreasing the maintainability of the project.

Study of Current Inception Frameworks

In this section, the main characteristics of the most relevant frameworks are going to be stated, as guidance for further comparison between them. Most of these frameworks are used to tackle the Agile Inception phase, allegedly setting the upcoming project in the right tracks.

Impact Mapping

Gojko Adzic proposes an Inception technique called “Impact mapping” [Adzic 12]. Its main goal is to assure that the assumptions that lead to a software solution are clearly and visually stated. This framework was created to support iterative medium-term plans, since it narrows the attention to a single outcome, typically one of high commercial interest.

Usually, business decision makers have a clear idea of the goals they want to achieve, but they may have trouble communicating them to others (e.g. development team). This lack of proper communication might lead to a series of inaccurate deliveries, making the project less effective or even a failure.

Levering Unambiguous Communication

In order to avoid common problems in Agile software development as: scope creep, pet features, and so on; caused by the lack of unambiguous communication, Gojko uses his impact maps to lay out visibly the WHY, WHO, HOW and WHAT of the problems faced.

Why?

It is the main question displayed on the map, thus being in its centre. It represents the **goal** we want to achieve, the reason why we are doing this.

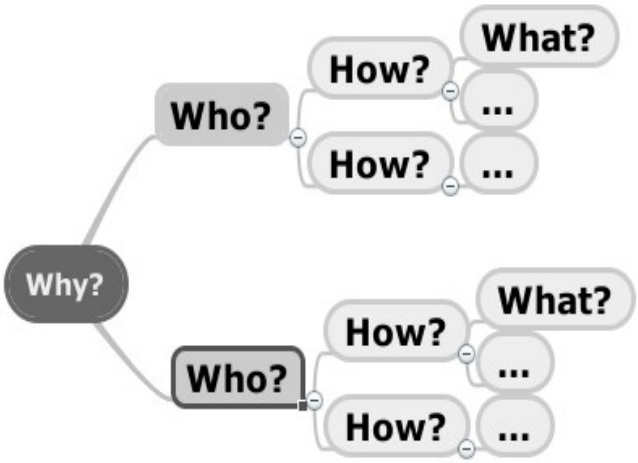


Fig. i Impact Map by Gojko Adzic

Who?

It represents the **actors** that will be influenced by the project or that could influence it. Here we wonder *"Who can produce the desired effect?"*, *"Who can obstruct it?"*, *"Who will be impacted by it?"*, *"Who are our consumers?"*. There are three types of actors: primary actors, whose goals are fulfilled; secondary actors, who provide services and off-stage actors, who have an interest.

How?

"How" sets the actors in perspective with our business goal. The answer to the following questions will give us the **impacts** we are trying to create: *How should our actors' behaviour change? How can they help us to achieve the goal? How can they obstruct or prevent us from succeeding?*

What?

It is the last question, it means the things we can do as a company, to make those impacts real. Therefore, we are talking about **deliverables**.

This visual representation helps us see the most relevant aspects of a project: goals, actors, impacts and deliverables; but more importantly, it helps us see the underlying assumptions that justify a branch of the map.

Fig. ii Impact Map Example by Gojko Adzic

The visual statement of the assumptions proves to be truly useful, since if business conditions change, it is easy to stop working in those branches whose assumptions do not longer hold.

Defining Requirements

Impact Mapping is a goal-oriented methodology. In other words, you will not be able to use Impact Mapping properly if you only have a set of features. You might try to extract business goals with something similar to the lean's 5-whys technique, asking questions until you get to the money. Remember that you have to get to something that shows how to earn money, save money or protect money.

Defining requirements as a set of features, instead of defining them to support usage goals and behaviour impacts, is one of the major reasons why IT projects fail [McManus 08]. This is due to the fact that communicating just *'what'* and *'how'* is to be done, does not allow team members to be prepare for unforeseen scenarios. That it is why, as of late, goal-oriented methodologies are gaining popularity.

Defining Good Measurements

Another key aspect for IT project survival, is to define a proper set of measures for each goal. Discussing these measurements leads to a more focused delivery. To establish precise measurements we need to take care of the following five items:

1. What is going to be measured.
2. How are we going to measure it.
3. The current situation.
4. The minimum acceptable value.
5. The desired value.

These items shall be specified for each goal we want to fulfil. It is recommend that summarized versions of the measurements appear on the map.

Identifying Key Priorities

Once the map has plenty of impacts, it is time to make some choices. This phase allows us to check the most relevant **impacts** (not deliverables) at this time. We should try to find critical aspects that can compromise the project, high-value low-hanging-fruit impacts or key assumptions to test.

If you are stuck, you might give your team some dots or virtual money to dot-vote impacts.

From Cost to Investment

Apart from proposing this framework, Gojko suggests a mind-set switch. Instead of seeing IT as a cost, he argues that IT must be seen as a investment. With a clear focus on business goals, which should be translatable to money, and good metrics, we can decide what the appropriate return on investment is for each goal, and if there is enough money, we can ask for a percentage of that budget to try to prove the assumptions that lead to that goal.

If the assumptions for a certain branch of the map does not hold, we will have spent just part of the budget. Afterwards, we will be able to re-think the strategy and make decisions easier.

Inception Deck

In his book, *The Agile Samurai* [Rasmusson 10], Jonathan Rasmusson defines what he believes to be the compulsory questions that must be asked at the start of an IT project. Since most projects are compromised by the assumption of consensus where none exists, Rasmusson created a 10-step Inception process called “Inception Deck”.

Fig. iii Assumption of consensus by Jonathan Rasmusson

To avoid this “assumption of consensus” pitfall, the Inception Deck aims at communicating the goals, visions and context of the project to the team, allowing them to make the correct decisions, and also, it aims at giving the stakeholders the information they need to make the decision of carrying with the project or not.

The idea is to start asking the tough questions from the beginning (specially at the beginning). In this exercise, the team shall spend enough time to solve the following questions with the people involved in the project.

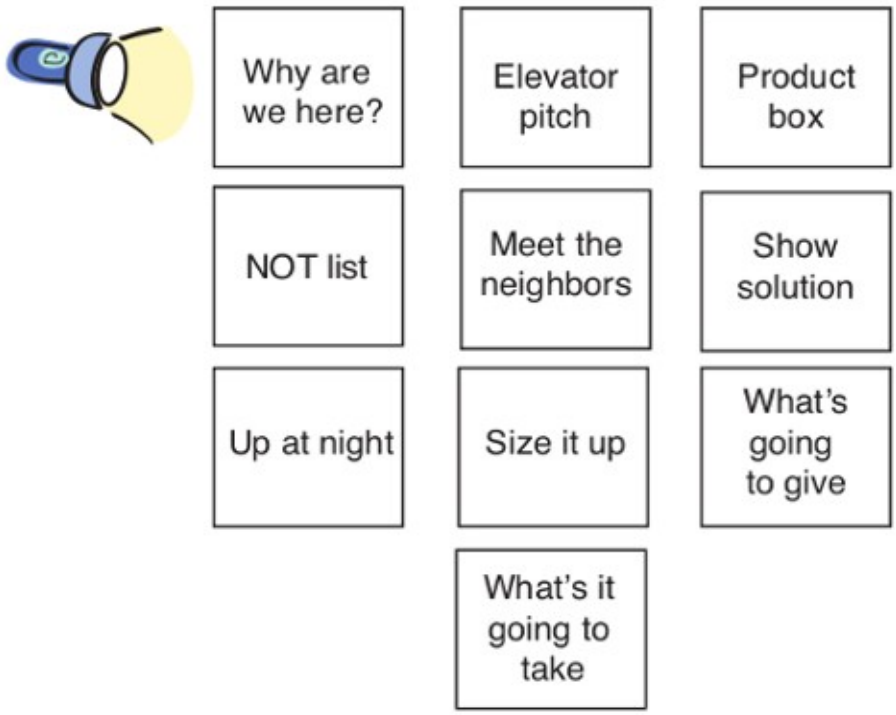


Fig. iv Inception Deck by Jonathan Rasmusson

Notice that Rasmusson says that his Inception Deck is a mere starting point, a set of guidelines which you can adapt.

Why Are We Here?

Knowing the reason why you are doing a project, is essential for its success. It all comes down to knowing what is the vision of the senior business decision makers (as the CEO) and to see for yourself why are you needed.

Most likely, the head of your company has stated a set of goals to pursue. This goals must be present when making decisions, thus, they must be present in this phase.

If you are able, it is important to literally see your customer needs, with your own eyes and those of your team members. This way you will truly get involved with your customers.

Elevator Pitch

An elevator pitch is a way of communicating the essence of your project, in just a pair of sentences in a brief window of time (usually thirty seconds to one minute).

This technique is broadly used amongst entrepreneurs, but a good elevator pitch brings also benefits to you and your team. An elevator pitch, forces you to think about the customer and makes you answer tough questions about what the product is and who it is for.

The suggested template for the elevator pitch is as follows:

- *For [target customer]*
- *who [statement of need or opportunity]*
- *the [product name]*
- *is a [product category]*
- *that [key benefit, compelling reason to buy].*
- *Unlike [primary competitive alternative]*
- *our product [statement of primary differentiation].*

Product Box

In this part of the process, the team will be creating a box for the software they are going to make, as if it was going to be sold at a store. This is a great team-building opportunity and a fun way to critically think about the *why* behind the project.

Firstly, you should brainstorm the benefits of your product, forget about the features, customers are only interested in how something is going to help them.

Secondly, you should create a slogan for your project. Do not worry too much, assemble your team for just ten to fifteen minutes in order to come up with something nice.

Finally, draw your box, give your team some colour pencils and star drawing a box to catch the eye of your customer.

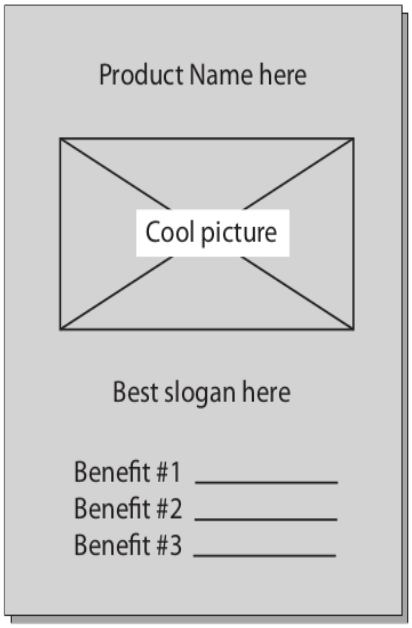


Fig. v Product Box by J. Rasmusson

NOT List

When defining the scope of a new project, stating what it is going to be about is almost as important as stating what it is not going to be about.

In this part, you should compose a list with three sections: **in scope**, for those features you want to focus on; **out of scope**, for those other things you are not going to worry for now; and **unresolved** which is the section for things you have not made a decision yet.

Meet Your Neighbours

Your “neighbours” is a fun way to refer to stakeholders, in a broad sense, or partners. Your neighbours in your project are all the people who might help you achieving your goals, from security auditors to database administrators.

In this phase, you should brainstorm with your team to get a list of people or companies with which you should interact before starting the project. Once you have a curated list, start talking about how to approach each group.

Show Your Solution

This phase aims at creating an overall technical architecture, nothing too elaborate, just enough for everyone to see how it is going to be.

With your technical team, decide how the project is going to be, from a technical point of view, this way, you set expectations around tools and technologies, visualize assumptions around project boundaries and scope, and communicate risks.

What Keeps Us Up at Night

When delivering software, your team will always face some challenges, thus, it is a good practice to discuss these possible challenges before they appear.

Then again, you and your team will most likely face unforeseen problems, but at least, you have shared with all members of the team most of the possible ones.

To carry on this part, brainstorm with your team possible problems and categorize them afterwards. You should end up with a list of problems about which you can do nothing to avoid them, so you should not worry about them, and another list of problems which you can prevent, so you need to start thinking on how to avoid them.

Size It Up

In order to present stakeholders with some estimates of when the software will be delivered, in this phase, you will make some rough estimate of the duration of the project. It is not hard to know if it is going to be a one-month, three-month or six-moth project.

Bear in mind that projects that last more than six months start to become risky. This does not means that you can not start projects which last more than six months, it means that you should break it down into more manageable pieces.

What is Going to Give

The first four sliders are the forces always present in a software project. As you slide them from left to right, you put less focus on them. They can not share the same level of importance, however, this does not mean that you will ignore some of these forces, all are relevant, the only thing is some of them might be more important to your client than others.

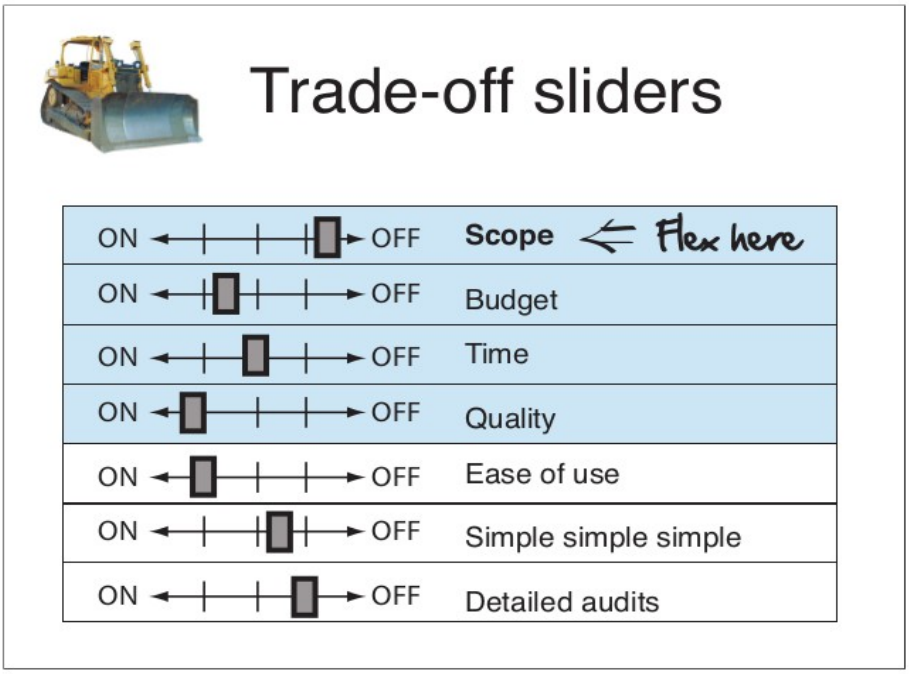


Fig. vi Trade-off Sliders by Jonathan Rasmusson

Although this forces are present in every software project, there are other relevant intangible properties that your team should consider. These other intangible properties are as important as the four forces, since a videogame on time and on budget, that is horribly boring counts for nothing. That is why, you should make your client clearly state them and prioritize them.

What is it Going to Take

In this last phase, you will present your sponsors with the team you will need and the money the project will cost.

At this time, you should have a more or less formed idea of how your team is going to be, however, once you start assigning roles and responsibilities, do not forget the role of the customer.

Also, in this phase you will need to identify or negotiate who is the leader amongst the stakeholders. This does not mean that you will not listen to the other stakeholders, but that they will need to talk to the team in one voice.

If you are required to give a rough budget, it is as simple as multiplying the salaries of the members of your team for the time it is going to take. Also you might need to add additional costs, as software costs and so on.

#	Role	Competencies/Expectations
1	UX designer	Capable of rapid prototyping (paper prototypes) Wireframes & mockups, user flows, HTML/CSS a plus
1	Project manager	Comfortable with ambiguity Can function without going command and control
3	Developers	C#, ASP.NET, MVC experience Unit testing, TDD, refactoring, continuous integration
1	Analyst	Comfortable with just-in-time analysis XP-style story card development Willing to help test
1	Customer	Available one hour a day for questions Can meet once per week for feedback Able to direct, steer, and make decisions about project
1	Tester	Automated testing experience Works well with developers & customer Good at exploratory testing

Table i What Is It Going to Take by Jonathan Rasmusson

Story Mapping

Jeff Patton proposes an Agile Inception phase called Story Mapping [Patton 14], aim at selecting the minimal set of stories for an useful release. This approach focuses primarily in this selection phase, rather than populating the backlog itself.

Collecting Features

As previously said, Patton does not give much guidelines on how to generate a backlog. Although, he does recommend that the list of features shall be user-centric. In other words, instead of describing an attribute or an object, you should try to describe an activity done by someone.

Once you have a list of features, it is time to add some additional information about them, namely, kind of user that will use the feature, its usage frequency and its value.

The outcome of this phase should be written down, in post-its or cards, since they will be placed in a board in an orderly fashion.

Feature Arrangement

The model to be used has two columns, the x-axis represents the usage sequence and the y-axis represents the criticality.

First, arrange the features across the x-axis in order to create a visual sequence of how things are done. It is not necessary that the sequence represents exactly the real word sequence, since there might be parallel tasks, you just need to order them in a way that seems logical when explaining the business process.

Once ordered by sequence, move the features along the y-axis as a group. The y-axis represent how critical is for your business the fact that someone uses a certain feature, in order words, you could think the y-axis as a frequency axis, where tasks highly used are at the top.

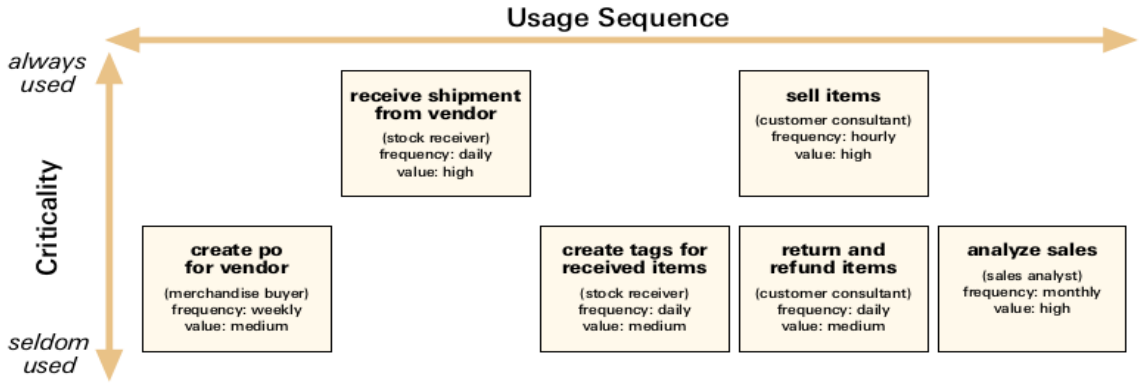


Fig. vii Story Mapping Example 1 by Jeff Patton

Finally, you should look for breaks in the logical work-flow. These breaks usually occur when there is a role change. When you have spotted this breaks, you should draw something like titled swim lanes, grouping, this way, the tasks that do not present these breaks between them.

Planning Releases

In this part, we are going to divide the model in system spans, which are a group of features that group together logically and cut through the business process, from start to finish. The first span should be the smallest set of features to be minimally useful in a business context.

In order to identify this first span, draw a line under the features of the top row, which are, indeed, the minimum group of features needed to make the system usable from end to end. This set of features will represent your first release (although, it may not be a public release). Creating this first release, will help the internal environment, such as testers or software architects, to start validating and testing things out.

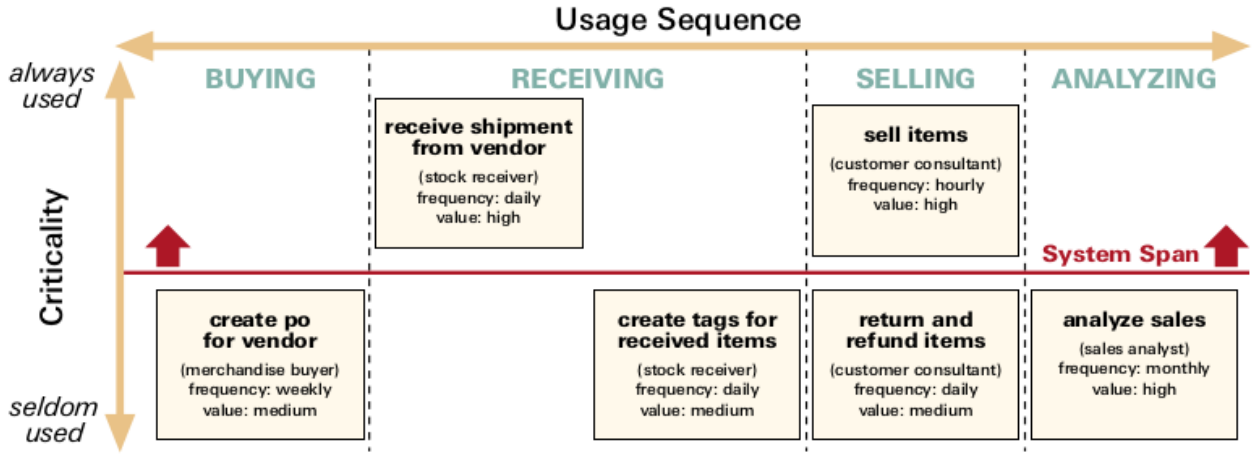


Fig. viii Story Mapping Example 2 by Jeff Patton

Finally, work together with business people, to decide which set of features makes up the best release and draw further horizontal (or almost horizontal) lines across the board to plan future releases.

Kano Model

Although Kano Model [Kano 84], developed by Noriaki Kano, it is not an Agile Inception framework, it could be useful for classifying user-stories. Noriaki created this model to classify customer preferences, into five categories, to improve customer satisfaction when developing a product for them.

These categories do not have a fixed name, anyhow, I will be using the following ones:

- **Must-be Quality:** the lack of an attribute of this category causes certain dissatisfaction. On the other hand, the presence of them, does no increase the final satisfaction of the customers. These attributes are viewed as basic (e.g. bottle of water without leaks).

- **Two-dimensional Quality:** attributes under this category causes, as well, dissatisfaction when not fulfilled, although they cause satisfaction when they are.
- **Delighters:** these attributes only cause satisfaction when present, since they surprise the customer. This means that they do not cause dissatisfaction when they are not implemented.
- **Indifferent Quality:** the presence or absence of these attributes do not add up, nor diminish, the satisfaction of the customer.
- **Reverse Quality:** these category refers to a high degree of achievement. Since not all customers are alike, some might find that the product has too many features, resulting in dissatisfaction.

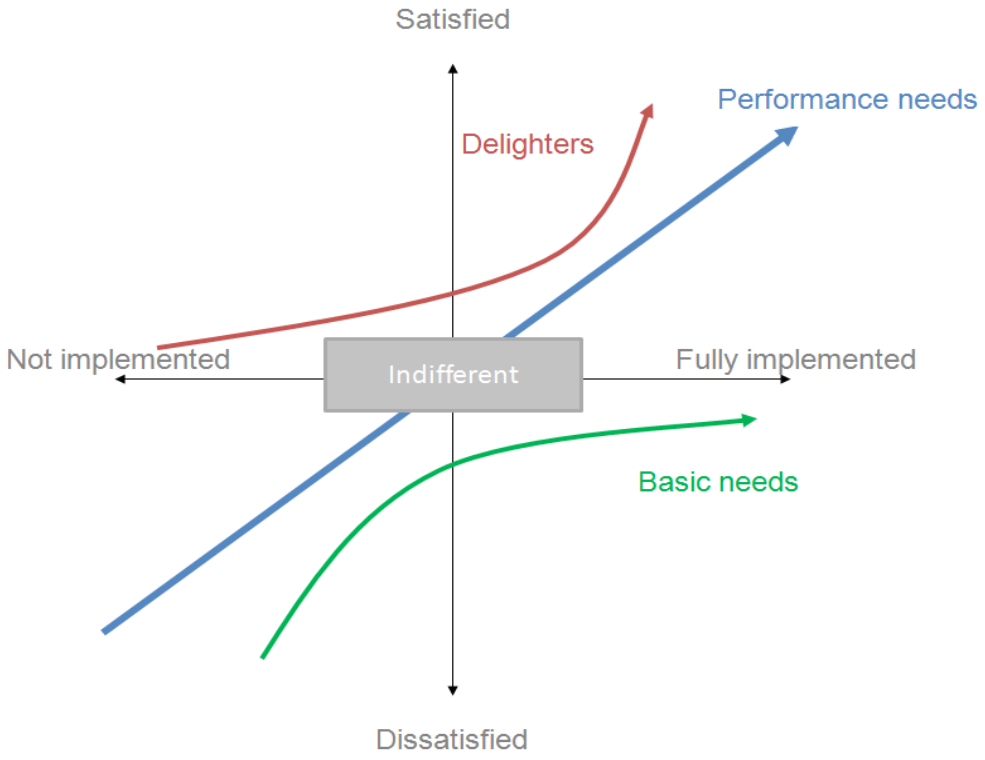


Fig. ix: Kano Model by Craigbrown

Bear in mind that, the category of an attribute will change overtime. Delighters might become performance needs and so on. This is due to the level of performance of competing products.

What is Agile Inception Then?

In order to be able to carry a proper study of the state-of-the-art of Agile Inception, I wanted to have a set of compulsory Inception features to judge each framework and each tool, under the same set of requirements.

The features presented in this section have been drawn from the previous analysis of inception frameworks, the author's personal experience and by thinking about how that experience might have been different, if some of those features would have been used.

At a high level, we could classify Inception characteristics into three different categories: user-story management, team synchronization and foresight.

User-story Management

User-story management does not refer, in this sense, to the handling of different stages of a certain user-story, but to the management of the creation and prioritization of these stories, as well as, selecting the right user-stories for a release. Within this section, I have included three Inception features, namely, product backlog conception, release planning and meaningful prioritization.

Product Backlog Conception

Properly filling a project's backlog, might not be so trivial as it might appear. It is common that, both the development team and the product owner, work together to fill the backlog with user-stories, but since they have, potentially, different backgrounds (e.g. Software engineers and Business managers), they might have trouble communicating their ideas.

Moreover, the way user-stories are written, might alter greatly the outcome. It is not the same to write user-stories without any order, than following a plan to compose those stories.

Release Planning

Choosing user-stories for a release based just on a ratio (e.g. priority/effort), might not be enough to satisfy the customer. We could be providing him with valuable pieces of software which are not related, delaying an end-to-end release, possibly damaging his business. The sooner we provide the customer with an end-to-end to show, the better.

Meaningful Prioritization

Using terms like “high priority” or “low priority” to classify user-stories, does not add much as to understanding what is their real value for the customer.

Moreover, those terms could be easily abused or confused by the customer, because *how low is low?*

Team Synchronization

Team alignment is of the essence if you want to have a healthy project, in which teams are productive and are allowed to come up with the best solutions for the problems faced. As Rasmusson stated, the assumption of consensus where none exists, is really damaging for a project.

In this sense, “team” is referring to an Agile team, not just the development team, since it utterly important that everyone shares the same understanding of what is being built. This category groups other three Inception characteristics: assumption revelation, end-to-end display and project rationale.

Assumption Revelation

Telling the team what to do is not enough to achieve the best solutions. If the people developing a platform, do not know why it has some specific characteristics, it is almost impossible for them to make well-founded decisions concerning its technical aspects.

Moreover, if the assumption for a set of user-stories is known, it will be trivial to cross out that set when an assumption does not longer hold.

End-to-end display

Visualizing in a simple manner the end-to-end value of an application, helps the team not only to plan better releases, but also it provides the team with a bigger picture of what it is being build.

Project Rationale

It is truly important to know what is that we are building, as well as, why is it necessary. The team must be able to know the answer to these question and it must be a unique answer, per question, for the whole team.

Foresight

It is true that Agile methodologies do not lose time trying to predict the future. However, this does not mean that we should remain oblivious of potential events down the road. Not making a brief study of future risks and the community surrounding a project, could greatly increase the costs of the solution.

Risk Discovery

Any software project has some associated risks. Exposing these risks early on, might prepare the team for what is to come, or could even help the team to mitigate that risk.

Community Discovery

As well as discovering the risks early on, it is also a good practice to know who is out there to help you. Apart from the obvious advantages of this, discovering your community might also expose requirements (e.g. internal standards) that went unnoticed.

3. Analysis of Current Inception Frameworks

Analysis Procedure

In order to fairly judge each framework, I compared them based on how many of the compulsory features of the Agile Inception phase did they cover. The more features a framework covers, the better.

The insights of each framework, needed to carry this analysis, where obtain from the previous section, in which the source of those insights were the authors themselves.

Result of the Analysis

√ √ √ Fully supports · X Does not supports · √ Partially supports

	Impact Mapping	Inception Deck	Story Mapping	Kano Model
Product Backlog Conception	√	X	√ √ √	X
Release Planning	X	X	√ √ √	X
Meaningful Prioritization	X	X	X	√ √ √
Assumption Revelation	√ √ √	X	√	X
End-to-end Display	√	X	√ √ √	X
Project Rationale	X	√ √ √	X	X
Risk Discovery	X	√ √ √	X	X
Community Discovery	X	√ √ √	X	X

Table ii: Inception Feature Coverage in Current Frameworks by Daniel Olea

Impact Mapping

Impact Mapping is a really useful framework for mature projects and for achieving business goals. However, there is a clear gap between the wholesome of a project and the areas covered by this framework. Using the example of an online game, Godjko presents a situation where the business managers want to get a million players, for this situation, Impact Mapping is very adequate, but how would you model the map for the creation of the game itself? This is why, Impact Mapping, can not be use for managing the entire inception phase.

Although it may not be used as a framework for the entire phase, one of its characteristics, stating the assumptions that support a certain task, is truly useful. This way, we avoid common situations in which the product owner wants something but explains something else. Also, we provide the development team with a bigger picture, allowing them to make the right decision about technology and tools.

Moreover, displaying this assumptions and their relationship with the user stories, allows the product owner and the development team to cross out user-stories that do not longer make sense due to an incorrect assumption.

In order words, adding this feature from Impact Mapping to the proposed framework, will let the team profit from the “assumption revelation” characteristic.

Inception Deck

The Inception Deck is a great framework, however some of the steps might not be so necessary. Rasmusson itself encourages to modify his Inception Deck to meet the needs of the project. In my opinion, the most important steps are: “Product box”, “Meet your neighbours” and “What keep us up at night?”. Of course, some of this steps require previous ones, but what I mean is that these three steps are the most valuable ones.

By doing a phase like the “product box”, the team will benefit from the Inception feature “project rationale”. Thanks to “meet your neighbours”, the team will experience the benefits of the “community discovery”. Last but not least, as of “risk discovery”, performing some step like “What keeps us up at night” will make the team profit from it.

Once again, Inception Deck covers part of the gap in the Agile Inception Phase, not the whole gap. Inception Desk is great for aligning the team, but it lacks some features that other frameworks have.

Story Mapping

Story Mapping is really useful when it comes to planning releases. This means that not only it will be easier for the team to plan what to release, but also that these releases will have a higher value for the customer.

Furthermore, Story Mapping allows teams to easily perceive the end-to-end value of the project, allowing them to see the big picture on the backlog.

Therefore, the adoption of these features will grant the team with the “release planning” and the “end-to-end display” Inception features.

Kano Model

This model, although it was not originally meant for Agile, could be useful to classify user stories. By using Kano's attribute classification, instead of using terms like “high priority”, “low priority” and so on, does not only standardize the classification of user stories, but also let the team understand more deeply, why a certain attribute is important for the customer. In other words, using this model will let the team profit from “meaningful prioritization”.

Conclusions

By reviewing the frameworks, it is clear that each one solves a specific gap in the methodologies, but none of the them solves them all. However, if we combine the different frameworks, we can see that all the features get supported.

In conclusion, I will use different parts of each framework to build the application. I will give direct support for the “Product Box”, “Meet Your Neighbours” and “What Keeps Us Up at Night” from Inception Deck within the application. This way the Inception features “project rationale”, “risk discovery” and “community discovery” will be supported.

Furthermore, I will try to merge Impact Mapping with Story Mapping to benefit from both of them. Story Mapping does not support the display of the assumptions underlying a user story and Impact Mapping does not help with the creation of releases, nor helps to see the end-to-end. This is why, combining both, could yield a lot of benefits since the application will be covering the following Inception characteristics: “product backlog conception”, “release planning”, “assumption revelation” and “end-to-end display”.

Finally, Kano Model will be integrated in the application as the standardize way to classify the importance of a user story, giving, this way, support for “meaningful prioritization”.

4. Agile Management Tools

In order to understand how supported the features of Agile Inception are in the existent Agile management tools, I have reviewed the most relevant ones, using the same criteria that was used to evaluate the frameworks.

Brief Description of Relevant Agile Management Tools

RALLY Software	
	Rally appears to be a mix between high-level sprint management and portfolio management. It seems that their target user are business managers, rather than the whole Agile team.
License	Proprietary

Table iii: Rally Software Analysis by Daniel Olea

Assembla	
	Assembla focuses in Agile software development, offering integration with SCMs (like Git), task and bug management and other features like basic team management.
License	Proprietary

Table iv: Assembla Analysis by Daniel Olea

Blossom	
	Blossom, at first, seems to be an Agile management tool by the book, then you discover they do not have backlog support...
License	Proprietary

Table v: Blossom Analysis by Daniel Olea

Jira	
	Jira is a team and project tracker, with many add-ons at hand. Some of that add-ons makes this tool a Agile management tool with backlog and sprint board support.
License	Proprietary

Table vi: Jira Analysis by Daniel Olea

PivotalTracker	
	PivotalTracker is another tracker with basic Agile support. However, the interface is so messy that it probably complicates things up.
License	Proprietary

Table vii: PivotalTracker Analysis by Daniel Olea

RedMine	
	RedMine is an open-source agile manager with SCM integration. It has support for the backlog, the sprint board, the burn-down chart and many more Scrum features.
License	GNU General Public License v2

Table viii: PivotalTracker Analysis by Daniel Olea

ScrumDesk	
	This is truly a Scrum management tool by the book. It provides support for all the practices present in Scrum. From “planning poker”, to “epic” support.
License	Proprietary

Table ix: ScrumDesk Analysis by Daniel Olea

ScrumDo	
	ScrumDo is similar to ScrumDesk, since they both cover a lot of Scrum features. Although ScrumDo seems to have fewer features than ScrumDesk.
License	Proprietary/GNU General Public License v2

Table x: ScrumDo Analysis by Daniel Olea

Sprintly	
	Sprintly has support for basic Scrum features and in addition, it present some other characteristics as of team management and progress tracking.
License	Proprietary

Table xi: Sprintly Analysis by Daniel Olea

Trello	
	Trello is not an Agile management software per se. However, with appropriate usage, could be useful to manage backlogs.
License	Proprietary

Table xii: Sprintly Analysis by Daniel Olea

Agile Inception Features Coverage

√ √ √ Fully supports · X Does not supports · √ Partially supports

	RALLY Software	Assembla	Blossom	Jira	Pivotal Tracker
Product Backlog Conception	X	X	X	X	X
Release Planning	X	X	X	X	X
Meaningful Prioritization	X	X	X	X	X
Assumption Revelation	√	√	X	√	X
End-to-end Display	X	X	X	X	X
Project Rationale	X	X	X	X	X
Risk Discovery	X	X	X	X	X
Community Discovery	X	X	X	X	X

Table xiii: Inception Feature Coverage in Current Tools (part I) by Daniel Olea

√ √ √ Fully supports · X Does not supports · √ Partially supports

	RedMine	ScrumDesk	ScrumDo	Sprintly	Trello
Product Backlog Conception	X	X	X	X	X
Release Planning	X	X	X	X	X
Meaningful Prioritization	X	√	X	X	√
Assumption Revelation	√	√	√	√	√
End-to-end Display	X	√ √ √	X	X	√ √ √
Project Rationale	X	X	X	X	X
Risk Discovery	X	X	X	X	X
Community Discovery	X	X	X	X	X

Table xiv: Inception Feature Coverage in Current Tools (part II) by Daniel Olea

Conclusions

The previous analysis serves to ratify the assumption that, nowadays, there is no single Agile management tool that covers more than two compulsory Inception features. There is, indeed, a market niche for an application that supports all these Inception features.

However, there are plenty of good Agile management tools that properly take care of the day-to-day of an Agile project. Therefore, besides being out of the scope of this project, supporting all of these other features of an Agile software development would be reinventing the wheel. That is why, we should select one of them and integrate our Inception-supporting application with it.

Since this is going to be an open-source project, it seems adequate that the integrated tool should be open-source too. This lets us with two options: RedMine and ScrumDo. In my personal opinion, after carefully reviewing them, ScrumDo seems to be more user-friendly and seems to cover more Agile practices. Therefore, ScrumDo will be the selected tool.

5. Framework Proposal: Agile Incepti-ON

From the previous studies, we have drawn the conclusion that there is a need for an Agile Inception framework that covers all its compulsory features. Therefore, in this chapter we will define the practices of this framework.

Product Box

This practice is exactly the same as Rasmusson's product box phase. Notice that, as Rasmusson suggests, before engaging this phase you should gather enough information regarding why the software being build is necessary and what is it that you are building.

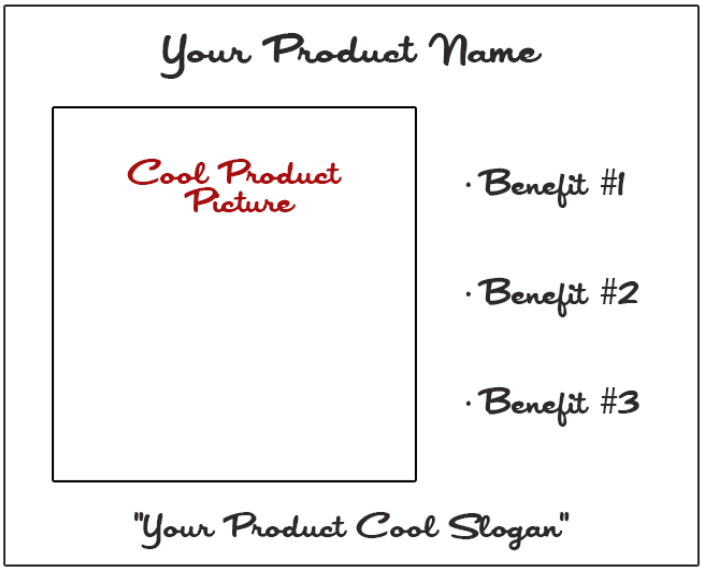


fig. iv: Product Box by Daniel Olea

The main objective of this phase is to provide the team with “project rationale”, which will be written down in a fun and simple way, assuring that all people involved share the same idea of the project, and also, allowing newcomers to understand what the project is about.

Project Battlemap

Creating a piece of valuable software is nothing short of a battle against the odds, but remember, Agile Samurais do not rout.

This practice covers the Inception features “risk discovery” and “community discovery”.

Firstly, begin to draw a square that will represent the battlemap. It is best if you and your team expend some time drawing trees and hills, that will improve the outcome of the phase as everything will be more fun. You might even want to print some real drawing of a battlemap. Be sure to create a map large enough to accommodate your risks and your community.

Secondly, draw two rectangles of the same size, covering the vast majority of the map, separated by a few centimetres.

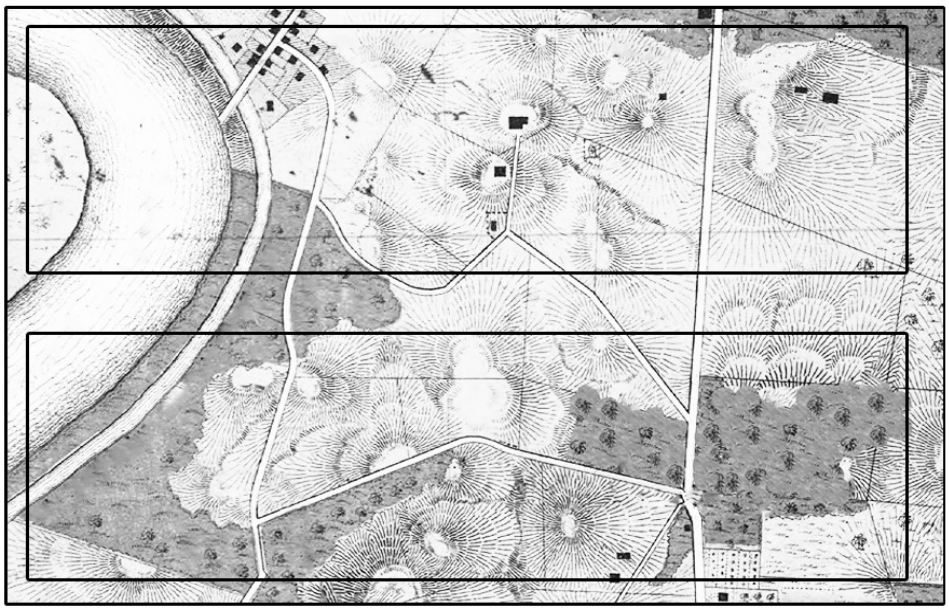


fig. v: Project Battlemap by Daniel Olea

Know your Foes

You will now identify, with your team, the foes of the project. The “foes”, are the risks you will encounter during your project. It is true that some foes will remain concealed, but try to discover as many as you can.

To carry this phase, start by brainstorming with your team potential risks your project face (e.g. customer availability) and write them down in a blank paper. When you think you have finished, discard those risks you can not do anything about (e.g. economic crackdown). Those left on the paper are you real foes; grab a red pencil a write them down in one of the rectangles. If you are merciful enough, you might give some formation to the enemy line.

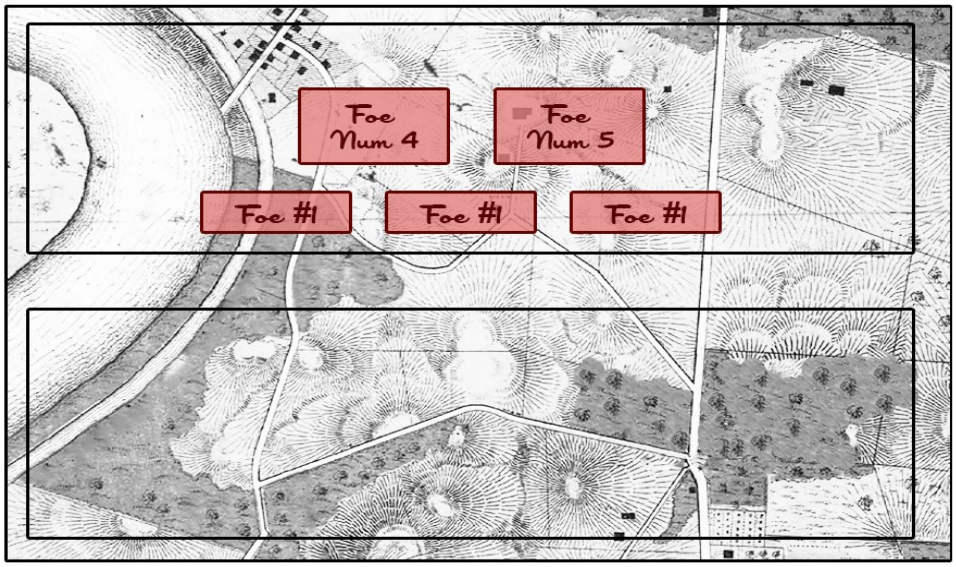


fig. vi: Product Battlemap with Foes by Daniel Olea

Meet your Allies

It is about time to call upon your allies! Again, brainstorm with your team what is the community that surrounds the project, and in what way they could be beneficial for your project (e.g. database administrators).

Once you have a complete enough list, use a green pencil to draw your team (choose collectively a name) in the middle of the opposing area, as close as possible to the front line. This is to remind you and your team that you are the front-line in this project, you might receive help from your allies, but the victory or defeat depends greatly on you.

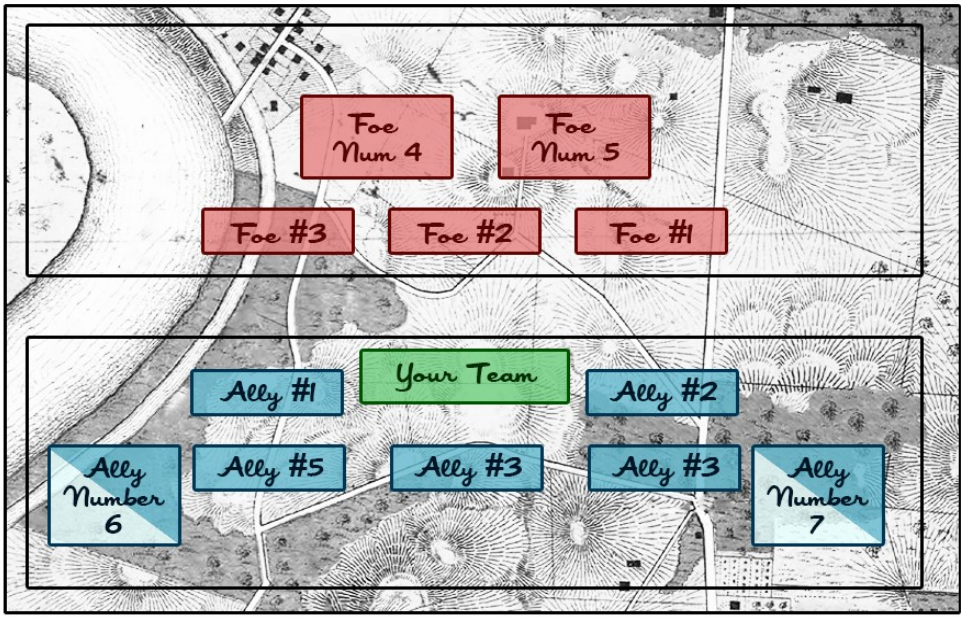


fig. vii: Product Battlemap with Foes & Allies by Daniel Olea

Now, it is time to place your allies in the battlemap, use a blue pencil for this. You could research some cool ancient battle strategies to deploy your army in a similar way, increasing the morale of your men.

Finally, discuss how could you prepare yourselves for the attack of each foe, or how could you defeat them before they bite. Also, discuss how to approach each ally in order to assure that they will be there when they are needed.

Backlog Devising

This final phase has a couple of practices. The objective of these phase is to cover all the Inception features left.

How to write user-stories

Bear in mind that this recommendation it is only meant for this early project state. It is inspired by Kano model [Kano 84] and Jeff Patton user-story writing [Patton 14].

Do not try to capture everything on the user-story, for now. Just write its title in a user-centric fashion (e.g. “add item to the kart”), and its classification as Kano suggested (deligther, must-be quality and so on).

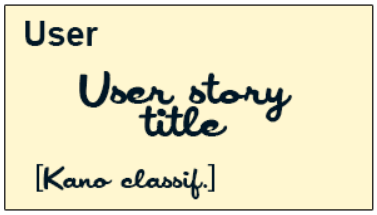


fig. viii: User-story Style by Daniel Olea

That will be more than enough now. Furthermore, by using Kano's classification model, you will achieve the “meaningful prioritization” feature of Agile Inception.

Money Upfront

In this part, you should discover what the business goals of your project are. In order to get to them, it is recommended that you follow Godjko's practices [Adzic 12].

Each business goal that you discover will be placed down in a board, as a titled swim-lane. It is recommended to create swim-lane two user-stories wide (e.g. the size of two post-its). Always draw these lanes with pencil.

It is common, that when you ask the product owner what the business goals of the project are, the answer would be something like “earning money by selling 'this' product”. That is not a bad answer, but it is too high level. Write that down as a single titled swim-lane that covers all the board.

Now you should discover with your product owner, sub-goals present in the project. Draw each one as a different lane. Each time you draw a new lane, ask the product owner what user-stories are inside that lane. In other words, what features does he or she think that will help achieve that business goal.

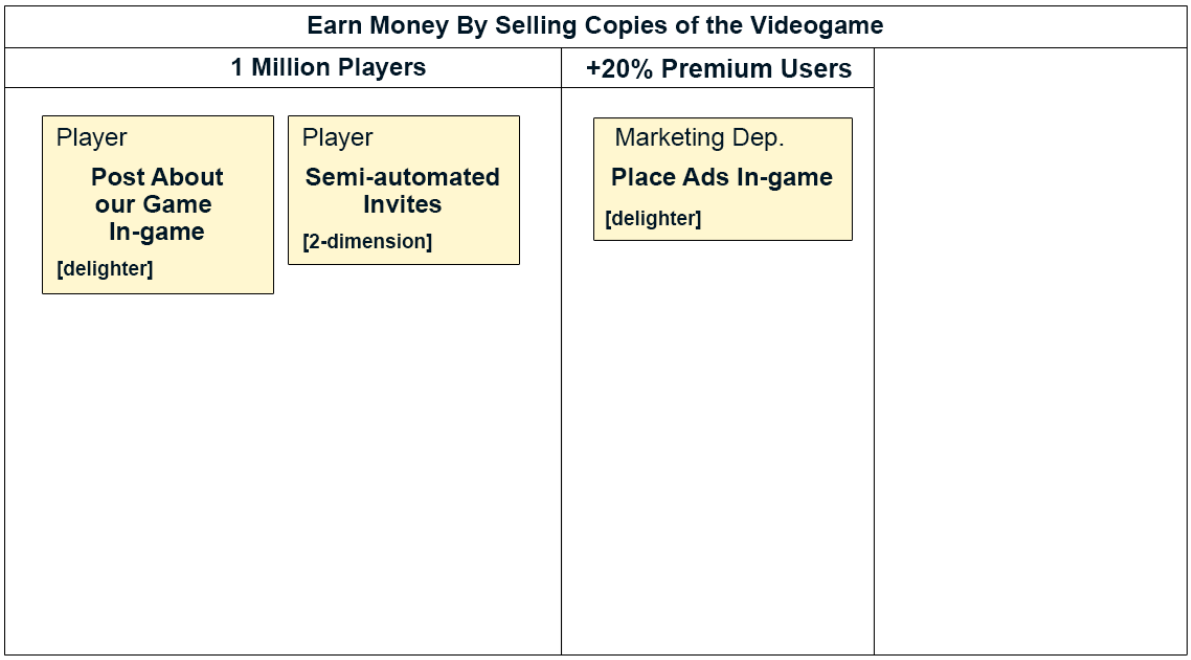


fig. ix: Discovering Business Goals Example by Daniel Olea

By carrying out this step, you will be uncovering the assumptions under a certain set of user-stories. In other words, you will now have covered the Inception feature “assumption revelation”, unlocking all its benefits.

Feature Placement

Not everything on your project will hang from a sub-goal, there will be plenty of the user-stories that will only hang from the root goal. Now, it is time to discover them. Try to start from one end and keep discovering features until you get to the other end. Do not worry too much for missed features, as they could be added afterwards (“We w*elcome changing requirements, even late in development.”).*

Notice that all user-stories are aligned at the top. This is necessary for a later classification of these stories.

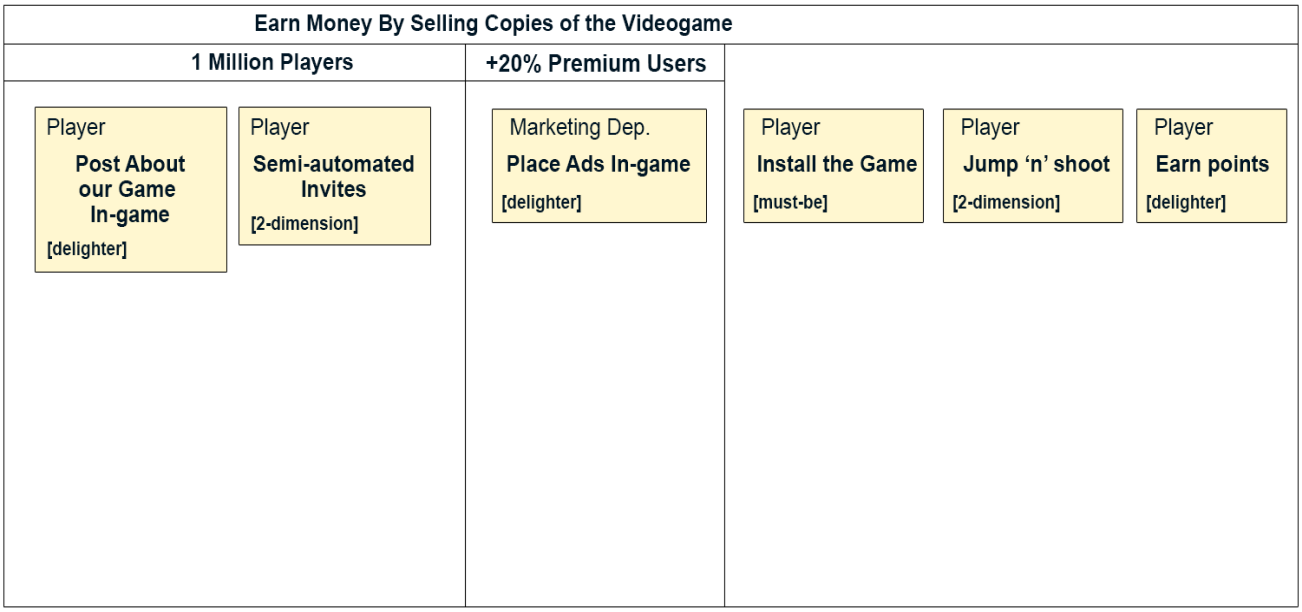


fig. x: Backlog Competition Example by Daniel Olea

Once you consider that you have covered all of the user-stories, it is time to arrange them. The user-stories should be placed in the same sequence as the will be used. Usually, if you have followed the previous end-to-end approach for composing those stories, great part of the work will be already done.

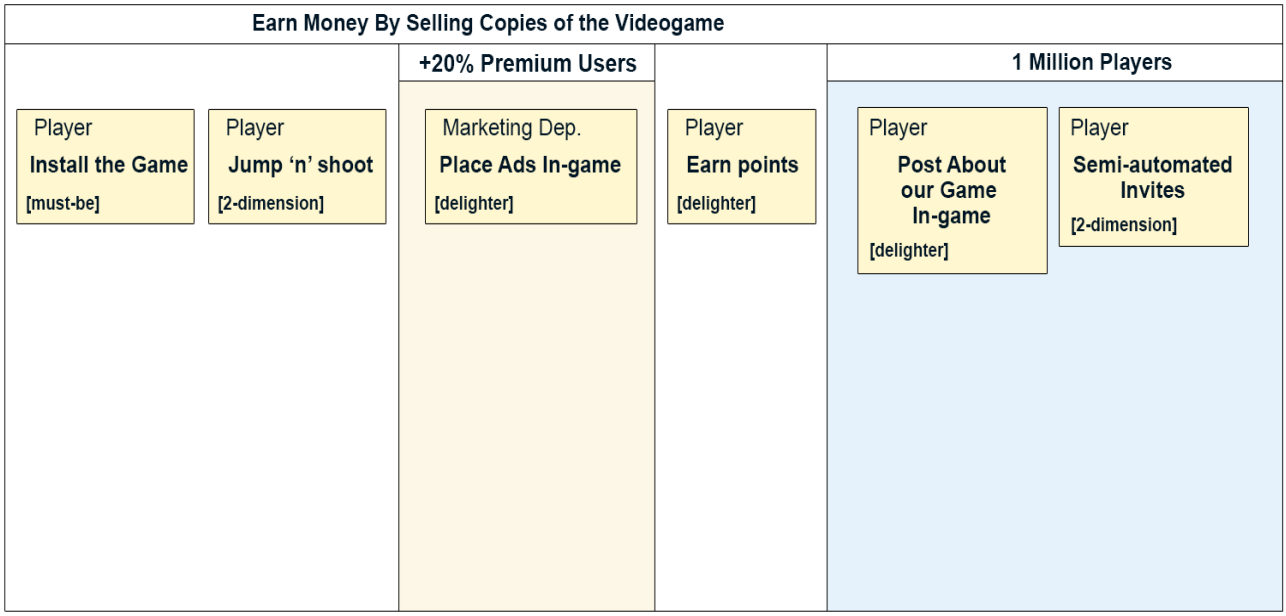


fig. xi: User-story Arrangement Based on Usage Sequence Example by Daniel Olea

Remember that the important thing is to have an clear end-to-end visualization, not an exact representation of the process.

If you see that two user-stories inside of a sub-goal are not properly ordered with the rest of user-stories, revise them, since that is not very common. However, you might have a rightful corner-case in front of you; if that is the case, feel free to duplicate the sub-goal in order to correctly arrange its user-stories.

Finishing this step will suppose that your team will be benefiting now from the “end-to-end display” Inception feature. Furthermore, this step compliments the previous one, which means that the team will also enjoy the benefits of the “product backlog generation” characteristic.

Creating the First System Span

This final step is the same as the Patton's one [Patton 14]. Now, it is time to vertically arrange the features. The most critical ones will be placed at the top, and the least critical at the bottom. Do not worry if this messes a little with the end-to-end order (you might specially experience this with parallel tasks).

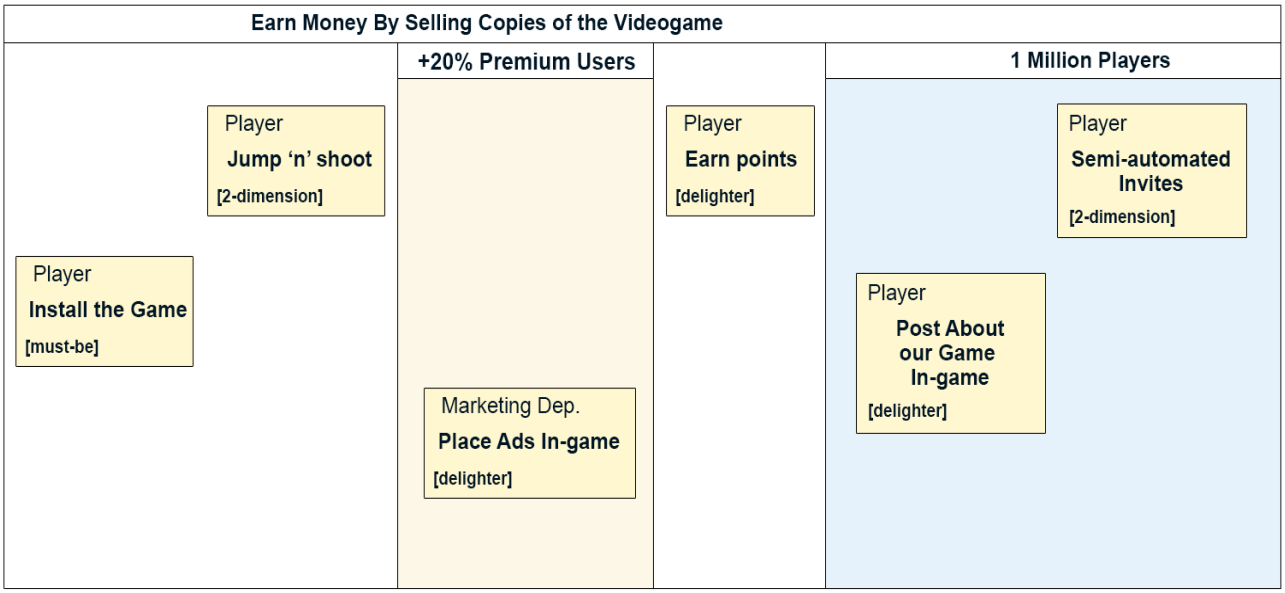


fig. xii: "User-story Arrangement by Criticality Example" by Daniel Olea

In order to identify this first span, draw a line under the features at the top row, which are, indeed, the minimum group of features needed to make the system usable from end to end. This set of features will represent your first release (although it may not be a public one). Creating this first release, will help the internal environment, such as testers or software architects, to start validating and testing things out.

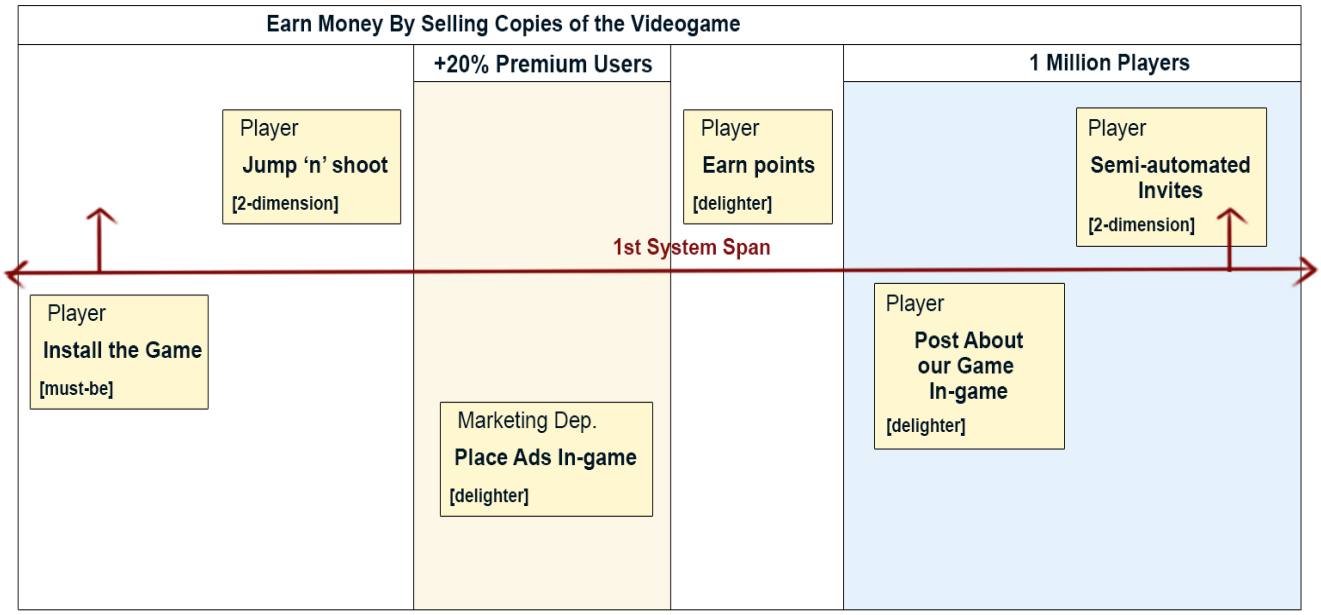


fig. xiii: "First System Span Example" by Daniel Olea

Now, your team will experience all the benefits of having achieved the “release planning” feature.

Conclusions

By carrying out all the steps of this framework, the whole Agile team will benefit from all the compulsory Inception features, greatly increasing the value of their deliveries.

√ √ √ Fully supports · X Does not supports · √ Partially supports

	Framework Name					
	Product Box	Project Battlemap	Money Upfront	Feature Placement	Creating the First System Span	∑
Product Backlog Conception	X	X	√	√	X	√ √ √
Release Planning	X	X	X	X	√ √ √	√ √ √
Meaningful Prioritization	X	X	X	√ √ √	X	√ √ √
Assumption Revelation	X	X	√ √ √	X	X	√ √ √
End-to-end Display	X	X	X	√ √ √	X	√ √ √
Project Rationale	√ √ √	X	X	X	X	√ √ √
Risk Discovery	X	√ √ √	X	X	X	√ √ √
Community Discovery	X	√ √ √	X	X	X	√ √ √

Table xv: Inception Feature Coverage in Agile Incepti-ON by Daniel Olea

6. Support Tool: Agile Dojo

Architecture

Architectural Style

As a starting point, I decided to follow a client-server architecture, the “3-tier” architecture specifically, because I believe that it is the most appropriate one for the project. Basically, I wanted to have a clear separation between the interface, the business logic, and the persistence due to the known benefits of this separation in web development. Moreover, experience has taught me that allowing only adjacent-tier communication, improves greatly the quality of the resulting code.

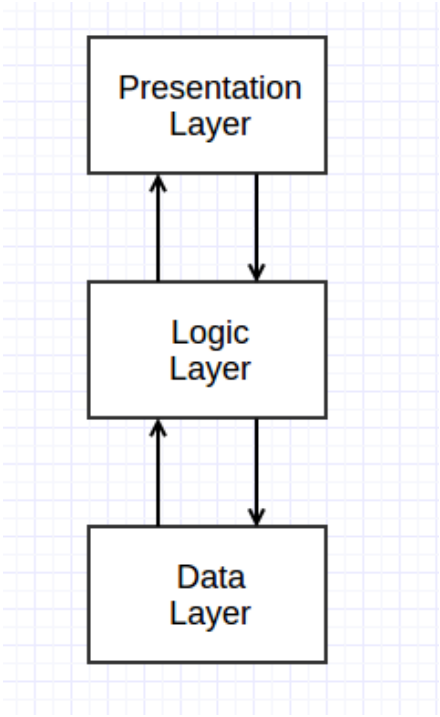


Fig. xi: 3-tier architecture by Daniel Olea

Notation

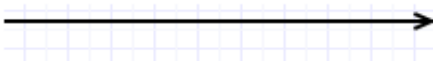
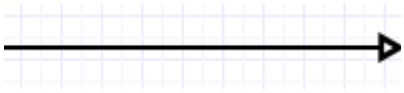
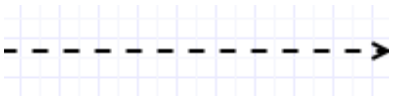
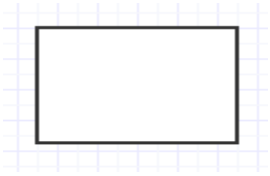
Symbol	Meaning
	*Usage*
	*Update*
	*Instantiation*
	*Observation*
	*Component*

Tabla xvi: Architectural Notation by Daniel Olea

Architecture View

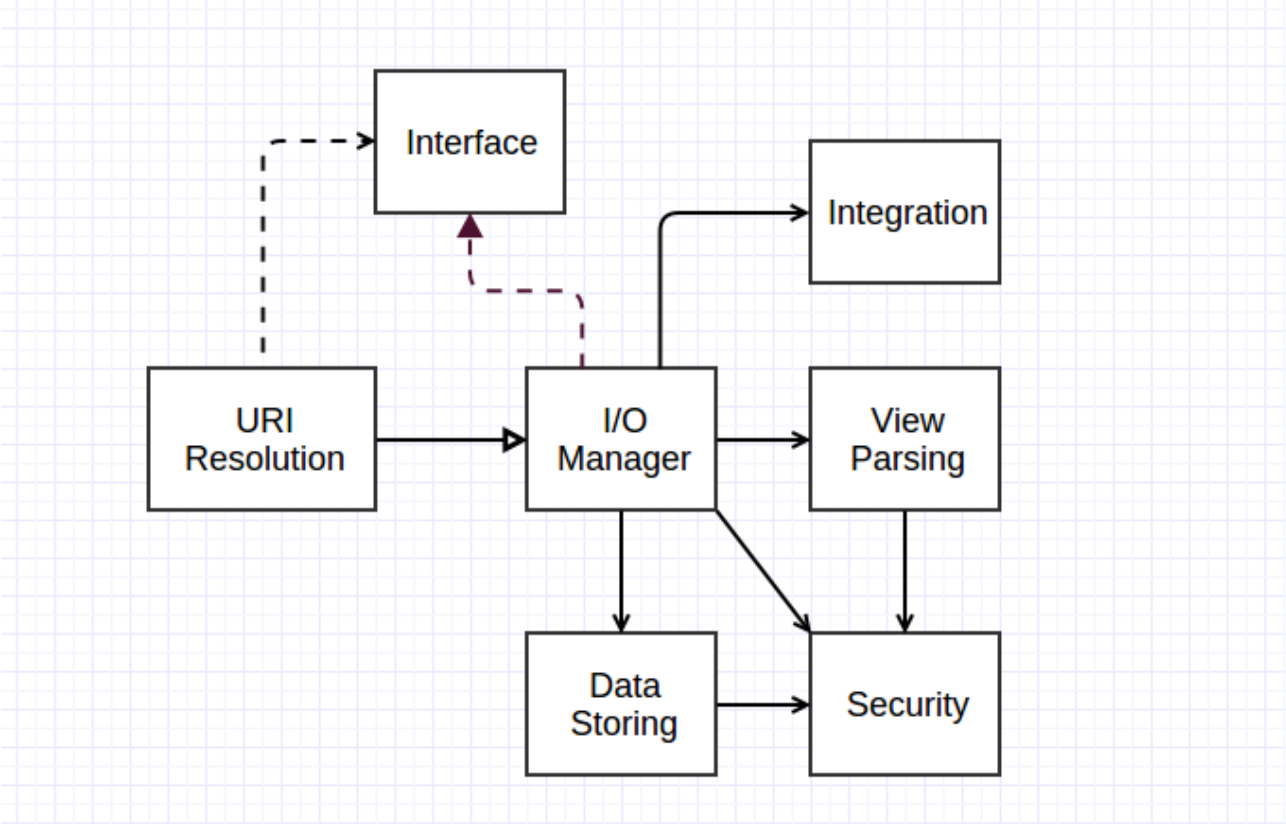


fig. xiv: Architecture by Daniel Olea

Rationale

- **INTERFACE**
  - This component category is needed in order to allow humans to interact with the application. It communicates with the logic tier through URIs, allowing unlimited sub-components of different nature to be plugged in. For example, a web interface and a desktop interface.
  - Its responsibilities are:
    - To display data in a comprehensive fashion.
    - To capture user interactions.
    - To invoke URIs.
- **URI RESOLUTION**
  - The “URI Resolution” component eases the communication between the user (through the “Interface” component) and the “I/O Management” component. As said before, this allows, potentially, any kind of interface to be use.
  - Its concerns are:

        - To observe the “Interface” enabling the communication with the “I/O Management” component.
- **I/O MANAGER**
    - This component holds a great deal of the business logic that depends on user input.
    - Its duties are:
        - To conduct the “Data storing” component and the “View parsing” component to provide with an adequate response to the user request.
        - To enforce a security policy using the “Security” component.
        - To integrate, when needed, the application with another Agile Software Management tool by using the “Integration” component.
- **DATA STORING**
    - This component represent the data tier of a 3-tier architecture. It is meant to store data without format limitations, e.g. files, text strings, and so on.
    - Its responsibilities are:
        - Persisting data as the “I/O Management” component says.
        - Returning to the “I/O Manager” the data asked.
        - Hiding the specific characteristics of the underlying repository.
- **SECURITY**
    - This component handles the security of the application.
    - Its concerns are:
        - To provide a diligent way of storing passwords in the repository.
        - To handle user authentication.
        - To provide resource access control.
        - To offer algorithms to avoid common web attacks.
- **VIEW PARSING**
    - Handles the creation of views to be returned to the “Interface” component.
    - It duties are:
        - Fill a view with the data given by the “I/O Manager”.
- **INTEGRATION**
    - This component takes care of the integration with other Agile Software Development tools.
    - It responsibilities are:

- Translate the data of the application to the format used by the other application.
- Communicate effectively with the other application.

How To Use Agile Dojo

This section aims at providing a quick overview of how to use the Agile Dojo's pages that give support to the practices of the Agile Incepti-ON framework.

Product Box Page

After logging in, you will see a screen with a single “create project” button. By clicking that, the process of Agile Inception will have started. The first step is the Product Box page.

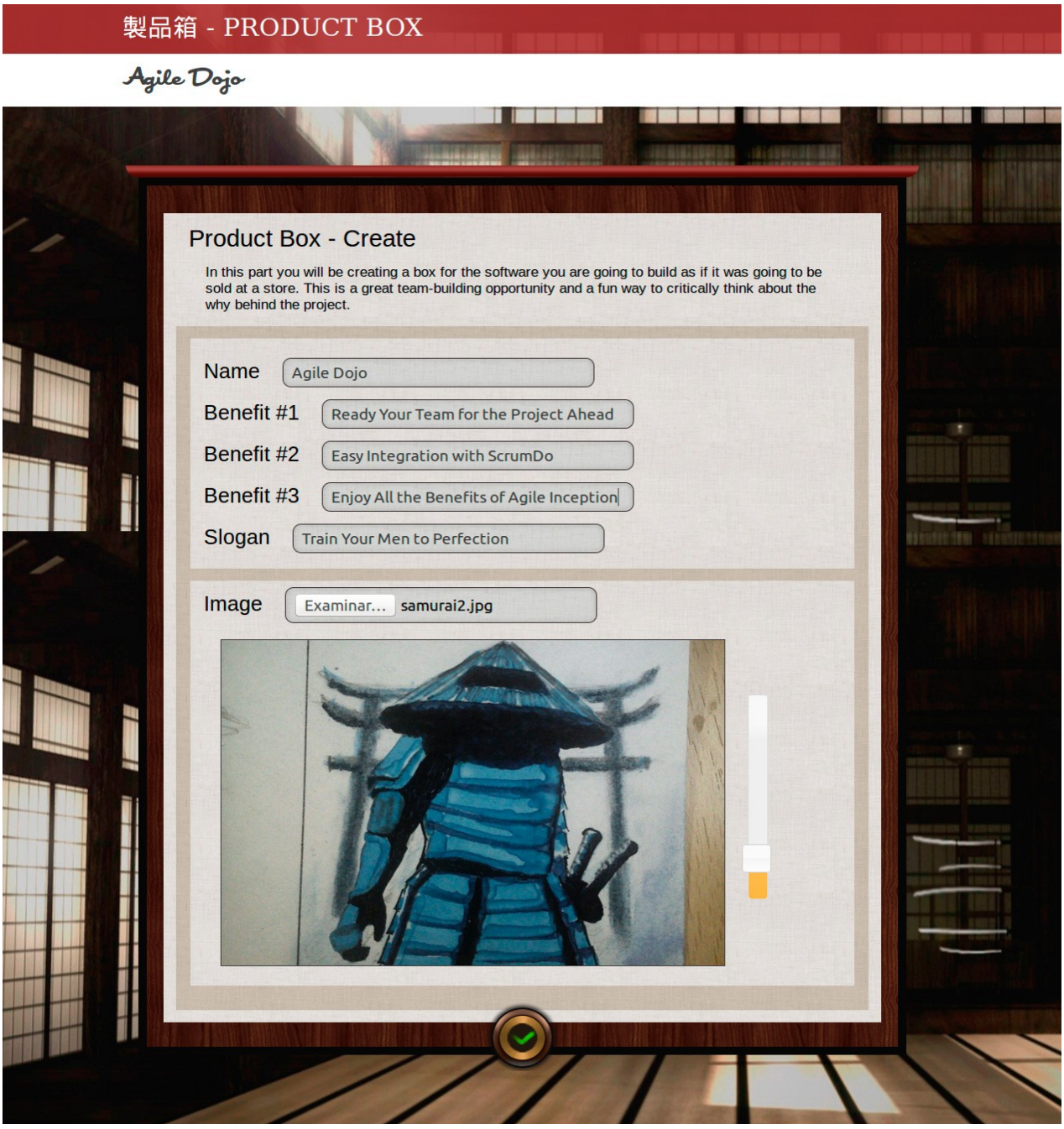


fig. xv: Agile Dojo Product Box (samurai drawing courtesy of Pedro Romero) by Daniel Olea

This page will let you write down the title of the project, its three main benefits and its slogan. All five inputs have a length restriction in order to make you synthesise their content.

In addition to that, the page will let you: upload the drawing of the product box, move that drawing within its container and zoom-in and zoom-out the image. Therefore, when you and your team start drawing the product box image, you do not have to worry about the aspect ratio or the size limitations.

The product box that you are creating now, will serve as an entry-point for the whole project (Product Battlemap, Backlog and so on). This way, every time you log-in in the application, you will be able to see the product box of the project.

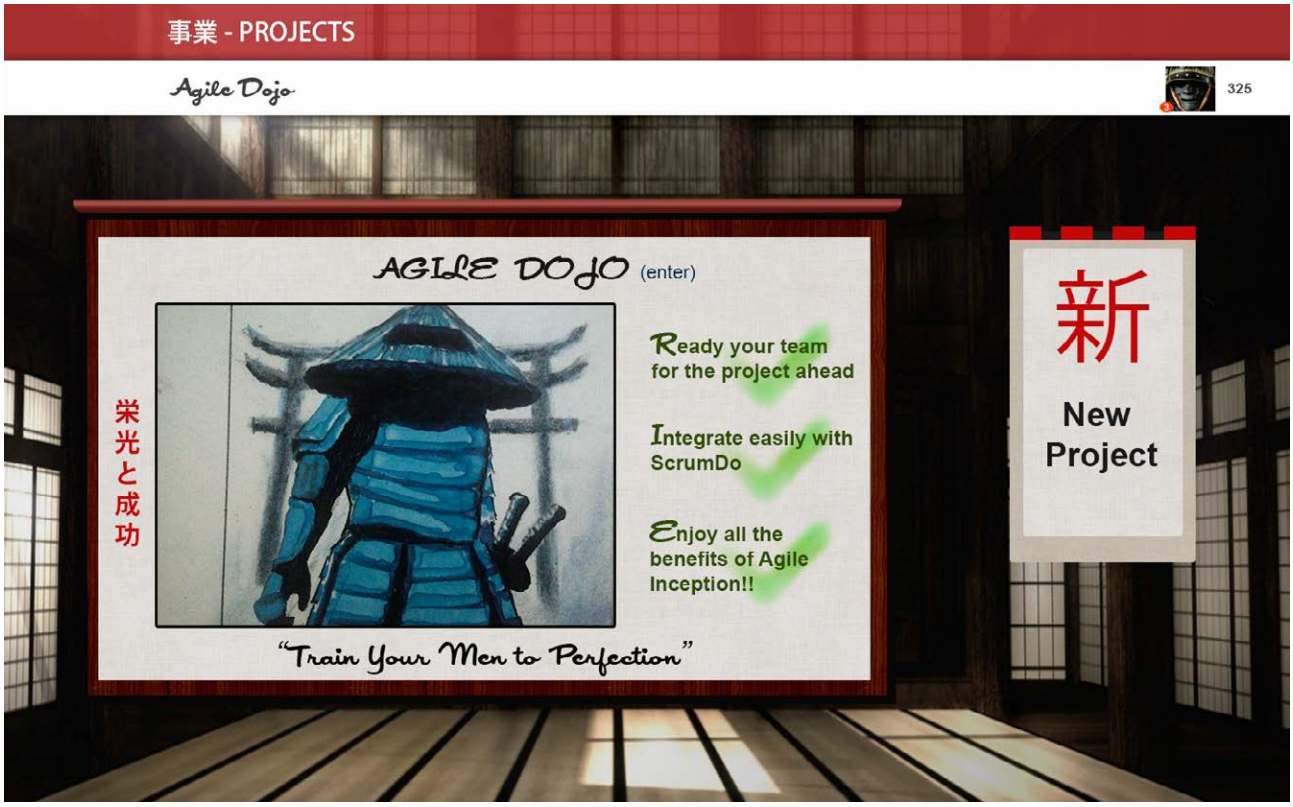


fig. xvi: Agile Dojo main page with Agile Dojo's Product Box (samurai image courtesy of Pedro Romero) by Daniel Olea

Once you click the gong, you will be redirected to the Project Battlemap page. Here, you will place both your foes and your allies, after brainstorming with your team, of course.

Project Battlemap Page

In this page, the battlemap feature is divided in two steps, the page, apart from providing a little tip about how to carry the process with your team, what I call “way of agile”, it will let you place your project's foes and allies by clicking on the rectangles of the map.

Know Your Foes Step

You can move around the foes, by clicking and dragging, edit their content, by double-clicking and delete them by clicking `Supr`.

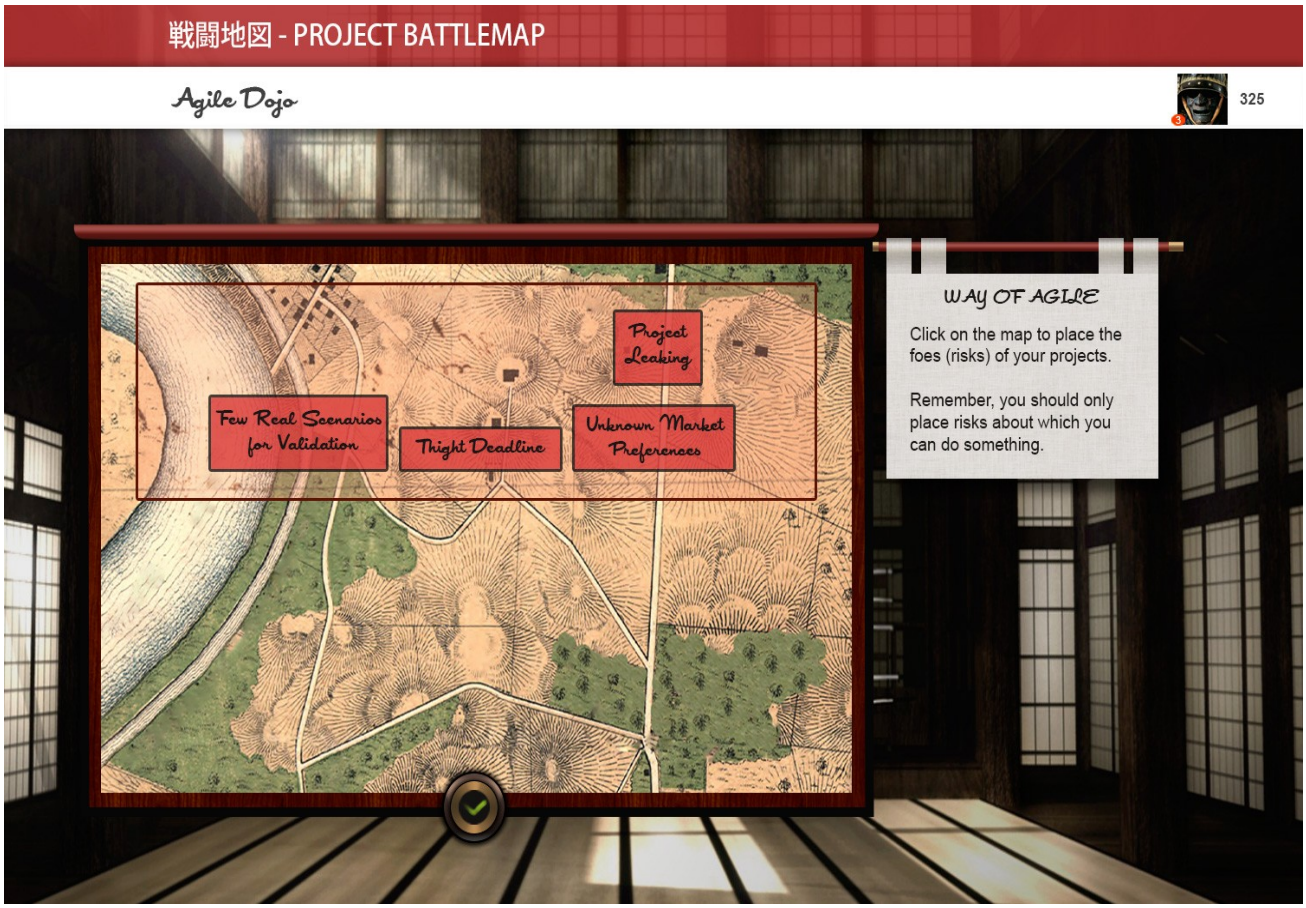


fig. xvii: Agile Dojo Project Battlemap with Agile Dojo's project foes By Daniel Olea

When you think you are done, click the gong. That will move you to the second part of the Project Battlemap. Now it is time to place your allies.

Meet Your Allies Step

In the centre of the ally zone, there is a green rectangle, that represents your team, choose a cool name for it and write it down, notice that this rectangle is the only one that you cannot move nor delete.

Now add you allies as you did with the foes, except that this time you will be clicking on the opposite area.

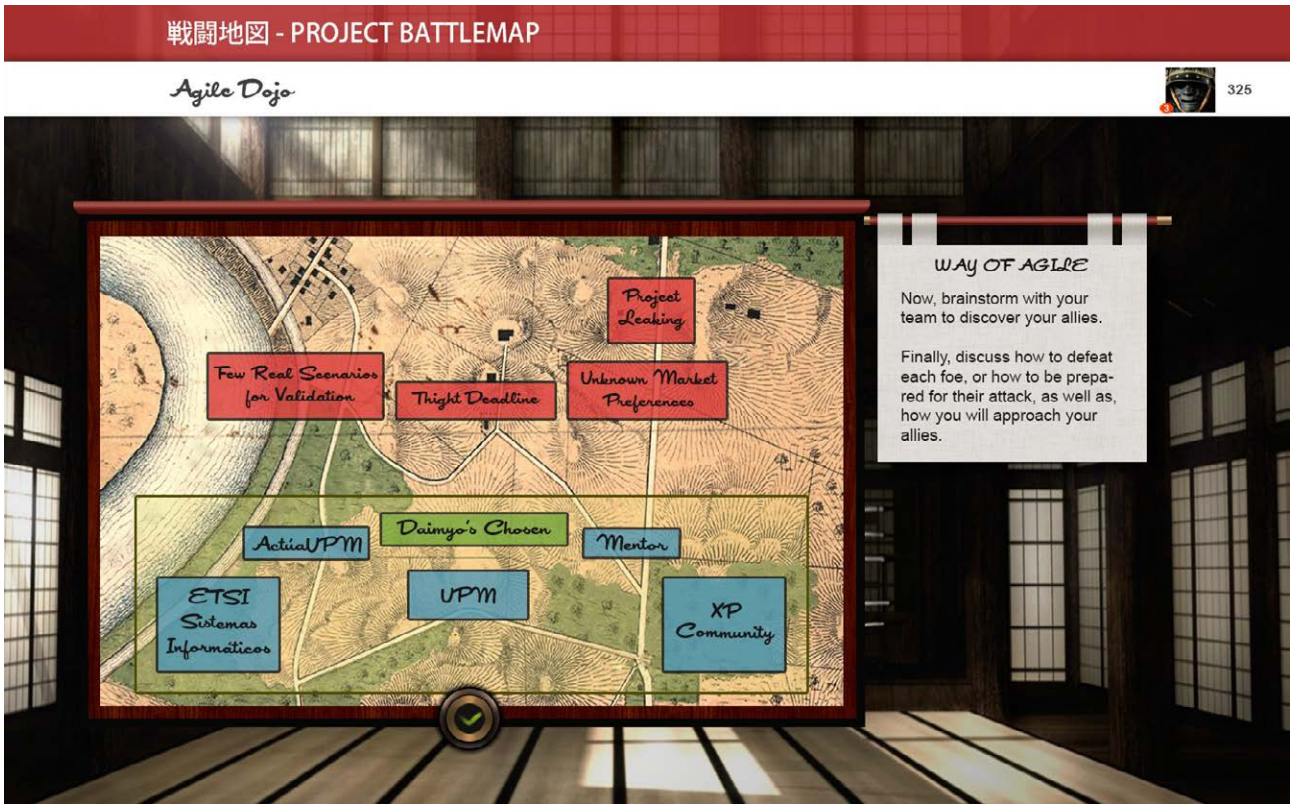


fig. xviii: Agile Dojo Project Battlemap with Agile Dojo's Foes & Allies By Daniel Olea

Backlog

Now it is time to start creating the backlog. This page, unlike the previous one, is only one step, but, remember that you should follow a process in order to achieve all the Inception features that we already know.

First, write the title of the root goal of the project. Then, start adding sub-goals, and their related user-stories.

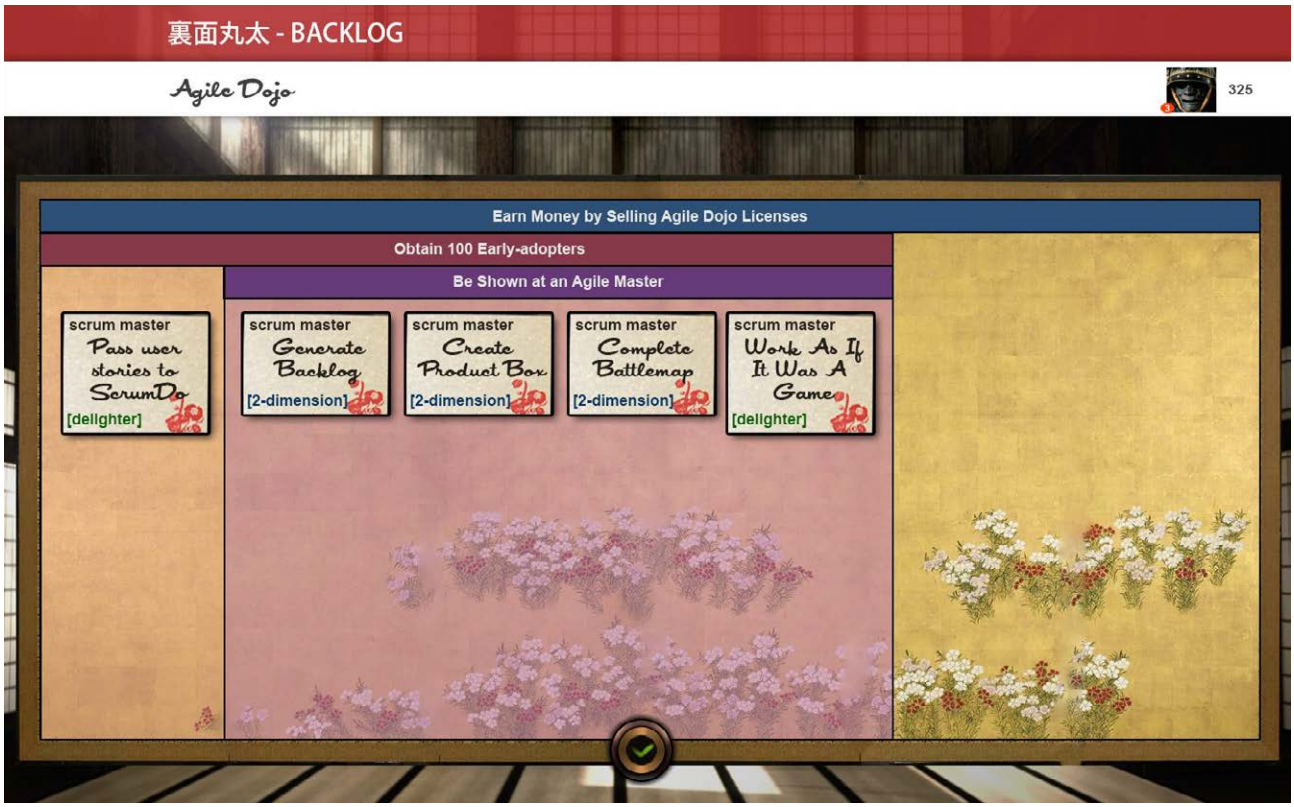


fig. xix: Agile Dojo Backlog with Agile Dojo's Business-goal Driven User-stories by Daniel Olea

Now, add the rest of the user-stories that are not motivated by a business sub-goal.

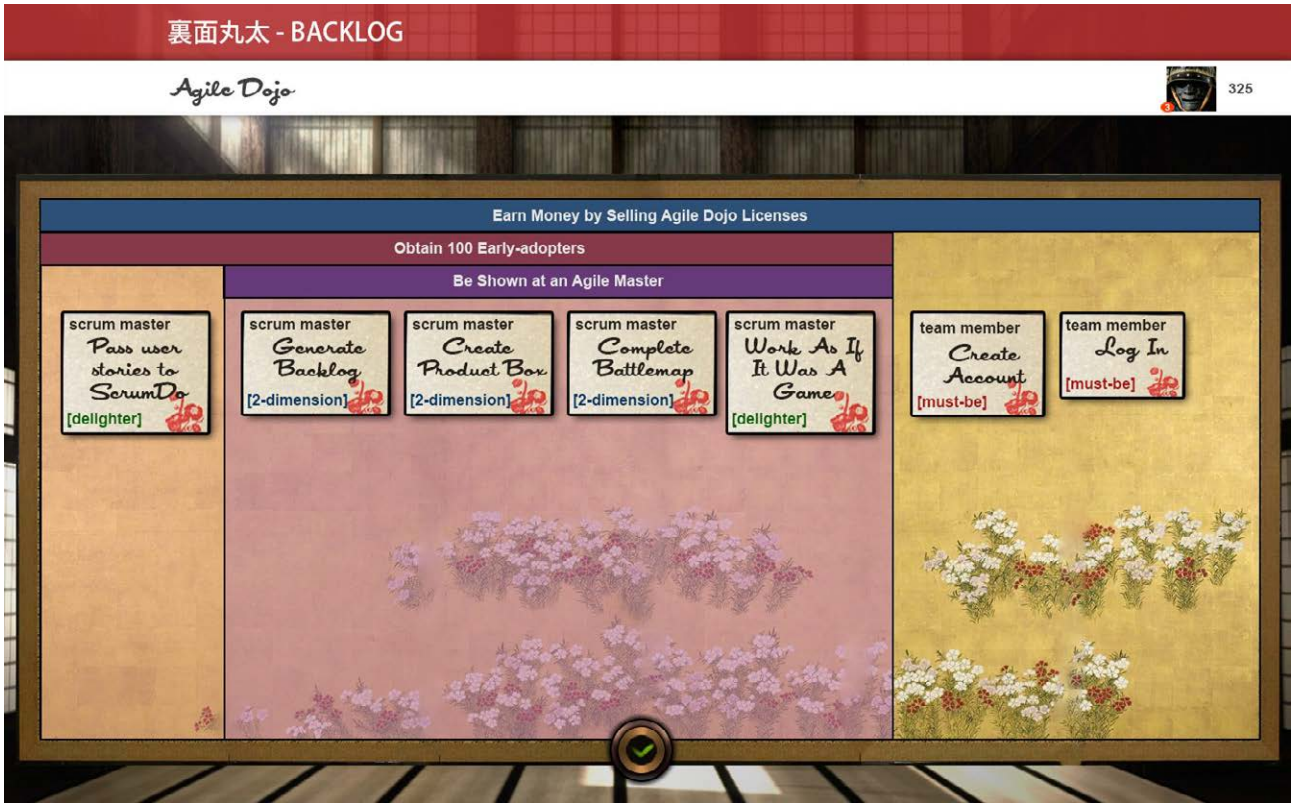


fig. xx Agile Dojo Backlog with All Agile Dojo's User-stories by Daniel Olea

Since you now have all the user-stories on the board, it is time to arrange them in the correct order as of usage sequence. In order to do this, click and drag business sub-goals' title to move them and all of their related user-stories, or click and drag independent user-stories. Remember that we do not need an exact order, but a clear conceptual order.

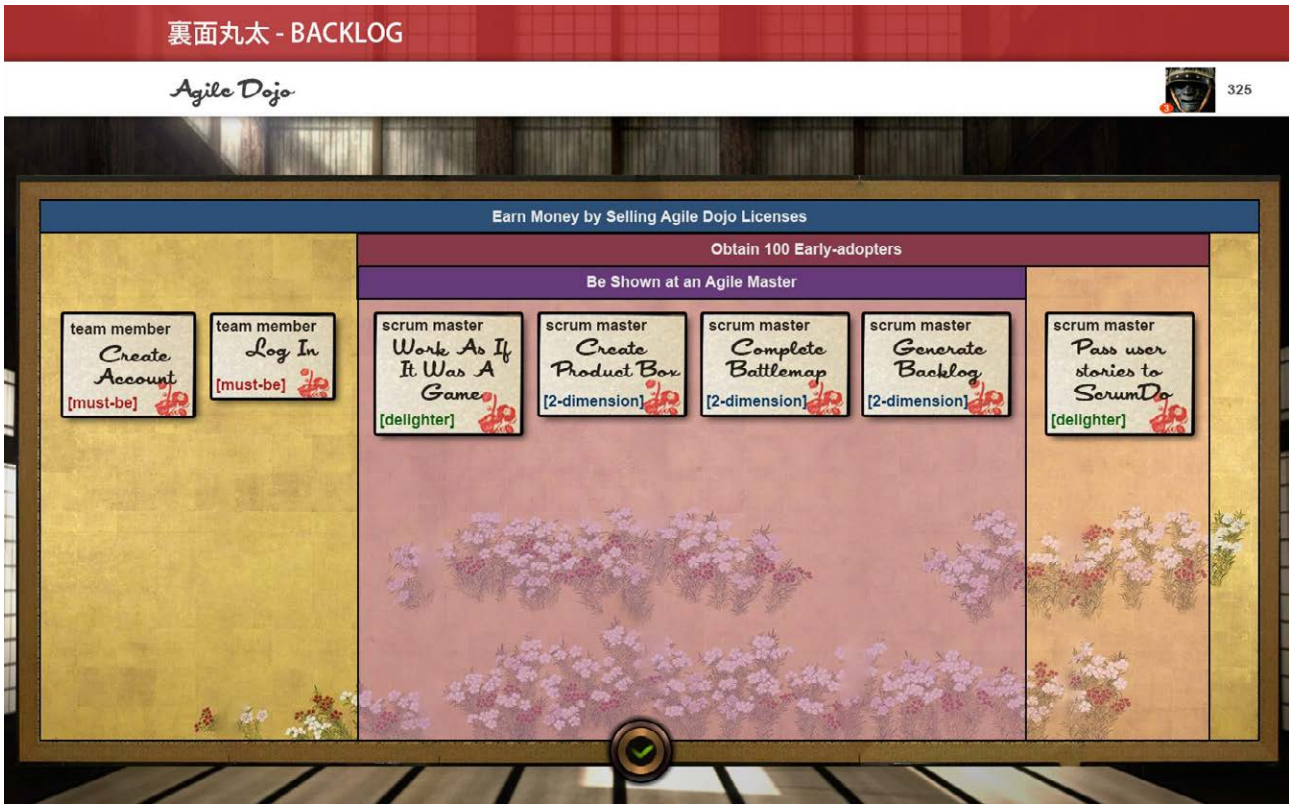


fig. xxi Agile Dojo Backlog with Agile Dojo's Arranged User-stories by Daniel Olea

Finally, order the user-stories around the y-axis by criticality. The user-stories at the top will be the ones with the most criticality, as oppose to those on the bottom, which will have the least priority as of now.

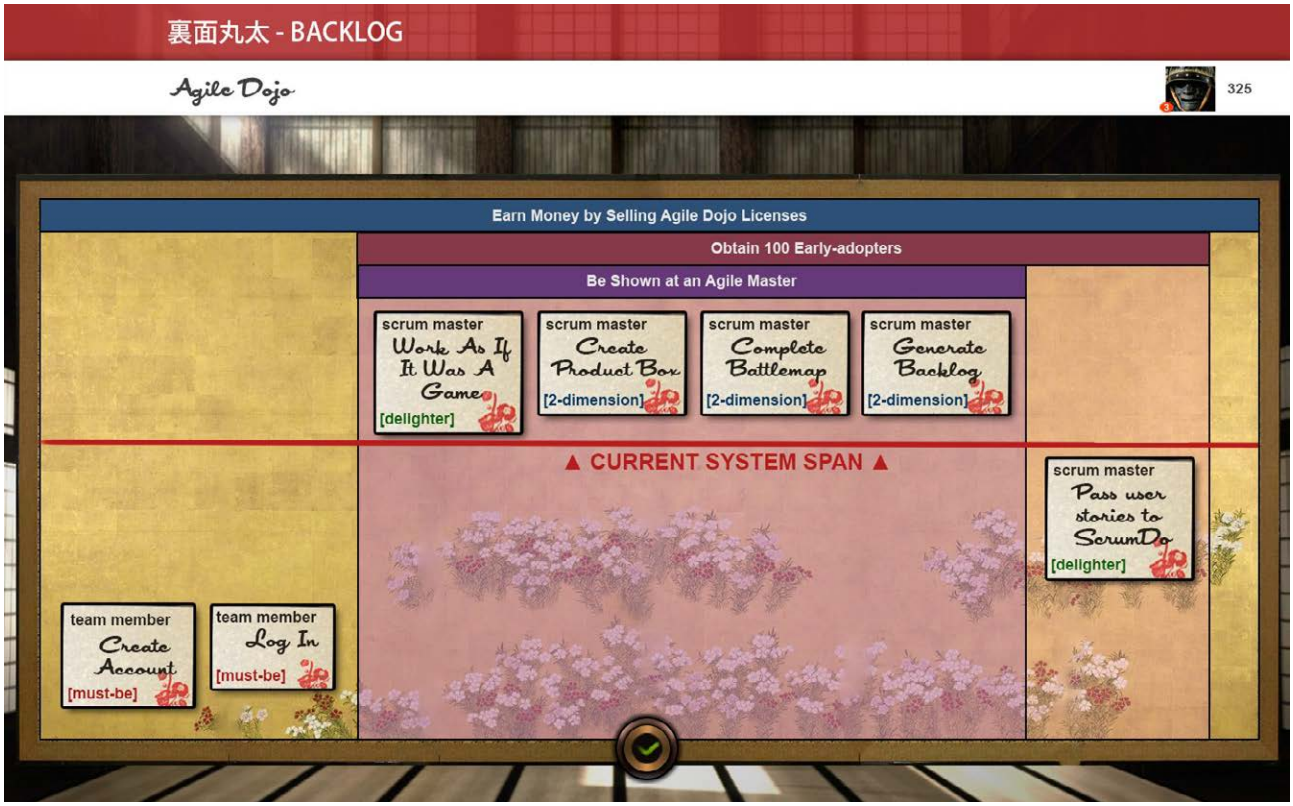


fig. xxii Agile Dojo Backlog with Agile Dojo's User-stories Ordered by Criticality by Daniel Olea

As you finish this part, you will have discovered what will be your first system span. If you like, you can click on the gong now in order to start the integration with ScrumDo.

This is integration is quite straightforward. The page will select those user-stories conforming the system span and will ask you if they are user-stories or epics. Depending on the answer, the page will let you compose one or more than one user-story. Some of the content will be already preloaded as it is already in the page, however, you might change it as you wish.

Mark the user-stories present in the backlog as finished as you complete them. When you complete all of them, re-prioritize, if necessary, the rest of the user-stories, bearing in mind that the new “top” is now just under the line of the first system span.

Repeat these steps as many times as you need.

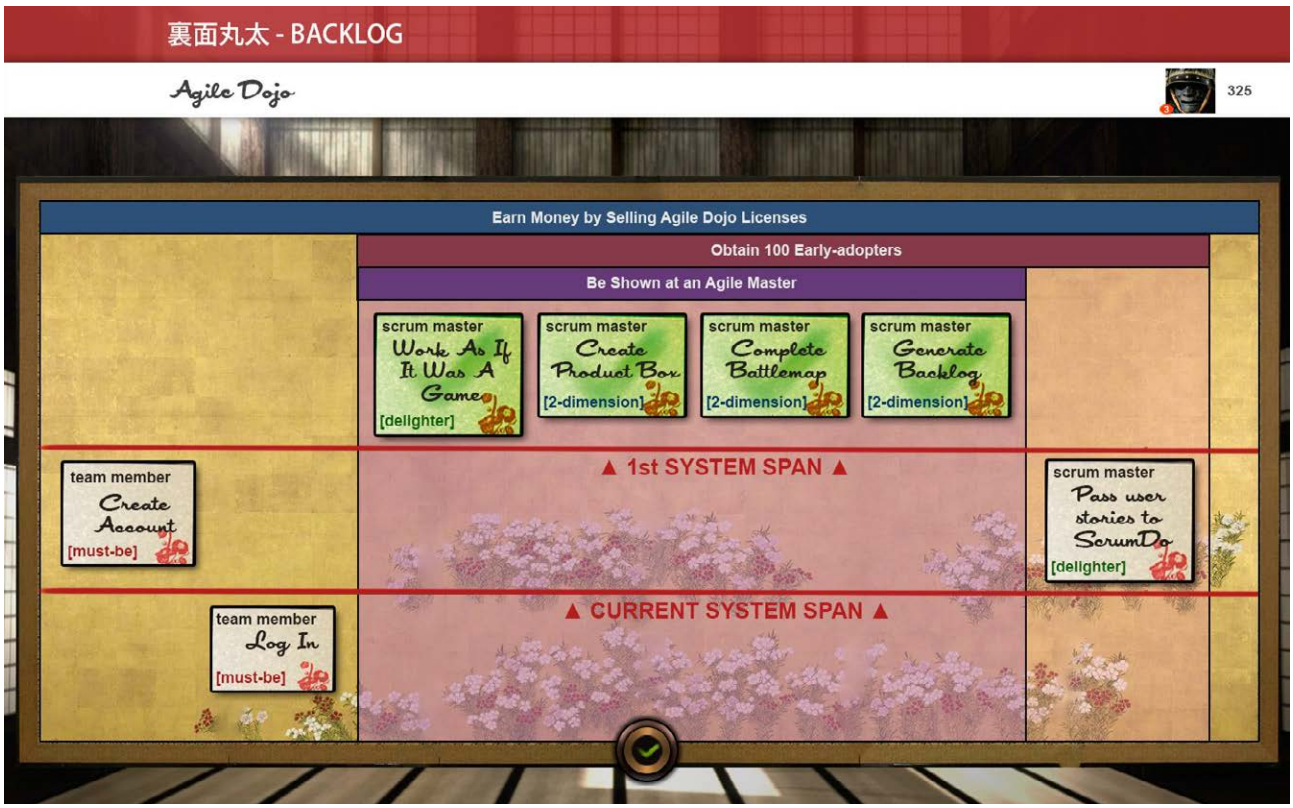


fig. xxiii Agile Dojo Backlog with Agile Dojo's Second System Span by Daniel Olea

7. Conclusions

Regarding the objectives of this dissertation, I can say that they have been fulfilled with diligence. The first objective, study of current techniques and methodologies of Agile Inception, was covered in the second and third chapters. The second objective, study of current tools for Agile Management, was fulfilled within the fourth chapter. Finally, the last objective, proposition of a framework to support the Inception phase and a tool to support that framework, was accomplished in chapters five and six, plus by the application itself.

The framework Agile Incepti-ON, proposed after a careful analysis of the state of the art of Agile Inception framework and Agile management tools, covers all aspects considered as compulsory for an Agile Inception phase. Furthermore, the practices devised to cover all those features of Agile Inception were created bearing in mind that a fun and simple process makes the team noticeably more productive.

In addition to that, the tool Agile Dojo was developed following similar guidelines of simplicity and amusement. I have to say that I am very proud of its current state. It covers all the features of the framework and also, it has a feudal-Japanese look-and-feel that adds a lot to the gamified experience.

As of Agile Inception , I can only say that I regard it as a compulsory practice, as long as you want to succeed with a software project. It helps you to align your team and to better understand the needs of your customer, clearing the path ahead.

Further Work

From this dissertation will rose a white paper for the sixteenth International Conference on Agile Software Development on May, 2015.

Needless to say, I will keep developing the application with future ideas. In addition to that, the tool Agile Dojo will be analysed to discover its business potential. If the result of this study is positive, Agile Dojo will become a spin-off from the university, engaging, probably, in the ActúaUPM competition.

Satisfaction

In order to conclude, I have to say that I am really satisfied with the results yielded by this dissertation.

8. References

[McManus 08] John McManus; Trevor Wood-Harper (June 2008). “A study in project failure”. BCS. The Chartered Institute for IT.

[Kano 84] Kano, Noriaki; Nobuhiku Seraku; Fumio Takahashi; Shinichi Tsuji (April 1984). "Attractive quality and must-be quality". *Journal of the Japanese Society for Quality Control* (in Japanese) 14 (2): 39–48. ISSN 0386-8230.

[KentBeck 01] Kent Beck; Mike Beedle; Arie van Bennekum; Alistair Cockburn; Ward Cunningham; Martin Fowler; James Grenning; Jim Highsmith; Andrew Hunt; Ron Jeffries; Jon Kern; Brian Marick; Robert C. Martin; Steve Mellor; Ken Schwaber; Jeff Sutherland; Dave Thomas (2001). “Manifesto for Agile Software Development”. http://agilemanifesto.org/

[Adzic 12] *Impact Mapping – Making a Big Impact with Software Products and Projects*, Gojko Adzic, 2012.

[Rasmusson 10] *The Agile Samurai – How Agile Masters Deliver Great Software*, Jonathan Rasmusson, 2010.

[Patton 14] *User Story Mapping – Discover the Whole Story, Build the Right Product*, Jeff Patton, 2014.